Collaborative Coding In The Cloud

Providing a paradigm shift to the way software development is achieved in environments of the future

Dominic Lovell 10304362

Honours Thesis

School of Computing and Communications

Faculty of Engineering & Information Technology

STATEMENT OF ORIGINALITY

I declare that the work presented in the thesis is, to the best of my knowledge and belief, original and my own work, except as acknowledged in the text, and that the material has not been submitted, either in whole or in part, for a degree at this or any other university.

Dominic Lovell

ABSTRACT

This research aims to address a number of challenges surrounding traditional software development practices, including the need for team transparency, synergy between project components and developers who are weighed down by desktop based environments. A cloud computing model is discussed, including a hypothesis on the required platform to solve many of these challenges. A number of previous research agendas are raised; including extensions to the JEdit and Eclipse IDEs, purpose built collaborative platforms, and an IDE that operates on a mobile device. Two cases studies around Google Wave and Mozilla Bespin are raised, and how industry leaders are addressing these challenges.

Through a qualitative survey, the needs of a developer and perceptions behind cloud computing are raised with a discrete range of industry professionals. A proposed model is provided, which aims at borrowing concepts traditionally found in social networking yet applies them toward a software development context, and highlights a number of recommendations for success. A research subset is then chosen to provide a technical implementation of a Google Wave agent, aimed at assisting distributed teams with cross communication and autonomous up-skill.

Finally, the research outcome answers the question of whether an IDE can be deployed within cloud based architectures and be adopted by the software development community. Given the infancy of the platform, the research outcome finds that immediate deployment of the proposed platform cannot be realized, and that researchers are dependent on platform maturity before successful deployment and adoption can be achieved. The overall research provides a number of future research directions, including reassessment of the philosophy proposed throughout this research, implementation of the proposed framework, or improvements focused on the communication and collaboration agent developed. The research fulfills a number of research areas required in the arenas of communication and collaboration among the software engineering community.

ACKNOWLEGEMENTS

I would like to extend my thanks to Chris Wong my honours supervisor, whose time I have occupied for the last twelve months in an already busy schedule. I would also like to thank Robert Castaneda whose inspirational direction gave me insight into my thesis topic. I would also like to thank Dion Almaer and Ben Galbraith for assisting with the distribution of my survey, and giving the world *Bespin*, the inspiration for my research.

TABLE OF CONTENTS

TABLE OF REFERENCE ITEMS

FIGURES

TABLES

1. INTRODUCTION

The following publication was developed to fulfill an honours thesis, and focuses on improving communication and collaboration among development teams. These improvements are facilitated through the power of internet technologies, and the emerging trend behind cloud computing.

Before discussing the key areas of this thesis, one must be familiar with the term cloud computing. According to Mark Hopkins of Mashable.com "'Cloud computing' has been used to mean grid computing, utility computing, software as a service, Internet-based applications, autonomic computing, peer-to-peer computing and remote processing. When most people use the term, they may have one of these ideas in mind, but the listener might be thinking about something else" (Hopkins 2008).

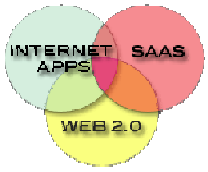


Figure 1- Intersection between areas of cloud computing

This thesis discusses cloud computing in terms of a merge between the concepts behind internet applications, software as a service (SaaS) and web 2.0 technologies. The publication assumes readers are familiar with these concepts.

The research aims to provide a framework that will support the improvements focused on communication and collaboration. A series of success factors will be outlined to promote user adoption, as well as a number of facilitators and obstructers that researchers must be aware of for future implementations.

This research will also look at existing literature behind these concepts, provide a hypothesis and rationale for cloud computing and look at a number of pioneering platforms. By conducting a qualitative survey, the research seeks to find why developers use their existing Integrated Development Environments. A technical implementation toward achieving increased collaboration and communication will also be addressed.

2. PURPOSE OF RESEARCH

The thesis titled *Collaborative Coding In The Cloud* aims to draw on three key concepts, those of, collaboration, development and cloud computing. Collaboration will be focused on providing contextual collaboration to the individual while also opening communication channels. Coding, apparent in definition, is concerned with software developers, software engineers, or those individuals that come from a technical background. The research will then specifically highlight the benefits surrounding cloud, or internet based, computing and the impact this has on the communication and collaboration needs of the individual.

The thesis aims to draw on existing research and highlight challenges surrounding software development, particularly focused on communication and collaboration among development teams. Through an assessment of popular architectures and infrastructures this research aims to evaluate the potential to utilize cloud computing, software as a service (SaaS) and web 2.0 patterns to provide developers with a transparent and collaborative development environment. Such an environment looks to provide a paradigm shift towards future software development practices and in doing so solves a number of management problems, primarily surrounding communication and collaboration.

Software development is inherently collaborative in nature. Business analysts must interact with customers to obtain requirements, and then work with system engineers and architecture to refine those requirements into technical implementations (Reeves and Zhu 2004). Programmers often work on the same piece of code, or requirements are modularized into discrete units of work, which are later brought together to form the unified set of work. Those developers, in order to work effectively, must be aware of their peers' operations, design decisions or tacit knowledge regarding the project. In a collaborative rich internet age, this workspace awareness should be up-to-the-minute knowledge about another individual's interactions and workspace (Greenberg et al. 1996). This collaborative process is referred to as contextual collaboration. According to

Cheng et al. (2003) contextual collaboration is 'an approach to collaboration in which users are not forced to leave their core applications to launch collaborative tools or visit a collaboration platform; instead, collaborative capabilities are simply available as components that extend standard applications' (Cheng et al. 2003, p. 21).

Over the last two decades, the Internet has made distances irrelevant and has made remote collaboration more practical among development teams. As management move development teams from single site locations to multi-site or globally distributed teams, many challenges rise particularly surrounding inadequate communication. Communication across multi-site teams supports dissemination of project knowledge, allows one to be familiar with the working styles of others in the team and fosters a greater understanding between team members (Sengupta et al. 2006). This distributed nature also presents a number of challenges relating to project coordination, peripheral awareness and collaborative coding.

Traditional software development environments are uncoupled from collaborative tools, and communication has been achieved through email, meeting notes and other asynchronous tools. This involves a substantial amount of context switching as users have to move back and forth between development environments and communication environments. This becomes difficult to track and preserve the discussions, requirements and design rationales that may be spread across several different media or tools. Developers may also need to look at software artifacts for information regarding technical implementations. However, this may be time-consuming, there may be ambiguities among artifacts, or the artifact may be obsolete. Therefore, there is a compelling case to provide developers with real-time contextual information regarding development activities and shared knowledge among the team.

This research will not only identify research issues surrounding these challenges, as well as suggesting a framework to address these challenges, but also looks to provide a research agenda for future implementations of such a framework. In doing so, the research aims to consider the implications on the next-generation of software

development tools. The broad aim behind this line of research is to outline the common collaboration facilities available on the Internet, and incorporate these into a development environment. However, the ultimate goal of this research is to answer the question of whether a collaborative Integrated Development Environment (IDE) can be successfully deployed using cloud based architecture.

3. PROBLEM, HYPOTHESIS, AND SOLUTION

3.1. CHALLENGES WITH EXISTING DEVELOPMENT PRACTICES

There are a number of challenges surrounding traditional development practices, as addressed by a number of sources, which will be outlined in the following section. These challenges arise from distributed team members and their inability to collaborate and coordinate effectively, a need for team transparency, and a lack of coherence between project components.

3.2. THE PUSH FOR GLOBAL TEAMS AND THEIR NEED FOR TRANSPARENCY

Traditional software requirements analysis and development rely on teams operating at the same location at the same time. These interactions largely occur face to face, or over a discussion in a meeting room. However, recent years has seen a push for globally dispersed teams operating over a number of continents. Organizations have benefitted from the greater access to highly skilled technical staff and lower production costs, yet in doing so have revealed a number of communication challenges they must face (Damian et al. 2006).

According to Olson and Olson (2000), geographic distance adversely affects the ability for distributed teams to communication. Through an empirical study, they argue that despite the use of tools such as video and phone conferencing, teams are still not aware of the context being delivered throughout the communication medium and a common ground between parties is often not achieved. They further highlight that their fieldwork displayed numerous examples where teams were unaware of the challenges they faced with current communication tools, drawing on an example that developers often adapt their behaviour to rather then fix the underlying technology. On many occasions they found participants would shout to convey the message only because they were unaware

that the volume on the remote site was set too low (Olson and Olson 2000, p. 154). This draws on the question of whether existing communication mediums are effective in allowing teams to communication and coordinate efficiently.

Herbsleb (2007) strengthens this argument by outlining that global software development teams face less communication, and the little communication they have access to is less effective when compared to they communication they produce at local sites. He attributes this to a lack of awareness and incompatibilities throughout the teams. The lack of awareness is marked by the fact that people at different sites share little context, and are unaware of what people at other sites are doing on a day to day basis, and thus do not share their concerns. This lack of context leads to difficulties in initiating contact, and often leads to misunderstanding of subject matter and motivation behind the work. Herbsleb finds that this ultimately leads to an inability to track the effect of changes as they propagate across to remote sites. While incompatibilities rise from differences in processes among the teams, which lead to confusion about how work is to be carried out, and the current status of the tasks.

Koskinen (2006) contends that despite the fact that people are sitting in different places, they should be able to interact and provide contextual information to each other about their aims, decisions, and actions taken throughout their work. Through the use of collaborative platforms, these individuals should have access to efficient and versatile interaction capabilities between teams.

An effect of these challenges has lead de Souza et al. (2004) to classify the need for social translucence, which they support with the view that there have been a number of recent research implementations that support the application of tools to support improved developer awareness. Such tools offer development activities to be visible through intelligent interfaces and integrated collaboration mechanisms within the environment. These tools can ‘adopt a different approach that focuses on constantly publicizing information’ (de Souza et. 2003, p. 112), which provides development teams with the transparency needed. A subset of these tools will be discussed in section 4.

3.3. A LACK OF SYNERGY BETWEEN PROJECT COMPONENTS

Many development practices offer a convention that allows work to be carried out independently, whereby each task is broken down into discrete modules to be later integrated to form the complete solution. de Souza et al. (2004) have found that such practices hinders a team's ability to collaborate, and often result in product conflict. They state that major problem faced by organizations who adopt distributed teams arises during the final integration period when the alignment between independent modules fails.

This process model teams often find attractive as work can be divided among the team and developers can specialize in specific areas. However, these workspaces 'create a barrier that prevent developers from knowing which other developers change which other [software] artifacts' (Sarma. 2003, p. 444), which later affects the integration between those artifacts.

This independent, often isolated, environment offers developers a lack of an appropriate awareness about other team members' activities (de Souza et al. 2004). This ultimately leads to a breakdown in communication and constitutes an apparent problem with existing collaborative environments. As this requirement becomes evident and the need for distributed teams increases, the need for a collaborative tool to support structured and unstructured communication and collaboration evolves (Cheng et al. 2003).

3.4. A REQUIREMENT FOR TEAMS ON THE MOVE

According to Belloti and Bly (1996), many studies around collaboration assume people conduct work at their desks, and such efforts have been directed toward development of new technologies that support collaboration from the desktop.

These desktop oriented tools include video conferencing, shared authoring and collaborative workflow systems. Belloti and Bly highlight that 'while these systems all have many benefits, they tend to be restricted to desktop collaboration only' (Bellotti and Bly 1996, p. 209).

Bellotti and Bly provide evidence into the need for increased developer mobility. Developers are often bound by cumbersome equipment, including laptop, peripheral devices and storage mediums to ensure they constantly have access to their development environment and related data. This presents a problem for developers who need to travel, or who require immediate access to their development environments. Such developers include those who constantly travel, have offices in multiple countries, or need to travel between multiple client sites.

Bellotti and Bly suggest that video conferencing and email cannot accomplish the communication necessary for mobility. As a consequence, they propose a solution for a portable device with wireless communication capabilities to provide context of developer locality, but then dismiss this notion due to its intrusive and expensive nature.

While the inability to easily port development environments is of concern, Bellotti and Bly state the principle challenge is focused around making local and remote distributed colleagues accessible to one another, and that a system that provides the presence of a developer would be advantageous to organizations.

3.5. SUMMARY OF CHALLENGES

Ultimately, the challenges outlined here are focused with five key areas. These encompass a lack of awareness across teams, a lack of effective communication channels, and inability to coordinate between teams, the need to locate individuals within the organization, and the need to be able to access a development environment from any location.

Given these challenges, research must be focused on providing a suitable platform to overcome these concerns. This platform aims to provide developers with an improved approach toward communication and collaboration procedures.

3.6. HYPOTHESIS

As the push toward cloud computing and software as a service increases, this leads way for an opportunity to address a range of business and software development problems. A cloud based environment offers a solutions where no downloads are necessary, no installations and upgrades are necessary for the end user, and there is no restriction on when and where development efforts can be completed.

Such an environment offers not only greater efficiency, but also opens but a range of new technologies and frameworks to be applied to a context that has yet to implement them. With this in mind, many development tools have moved away from desktop environment and have shifted toward cloud architectures, such as UML tools, wikis, issue trackers, and even source control tools such as Subversion and CVS can be hosted across internet environments. This begs the questions of whether an IDE can be deployed in the cloud, and be successfully adopted among the software development community.

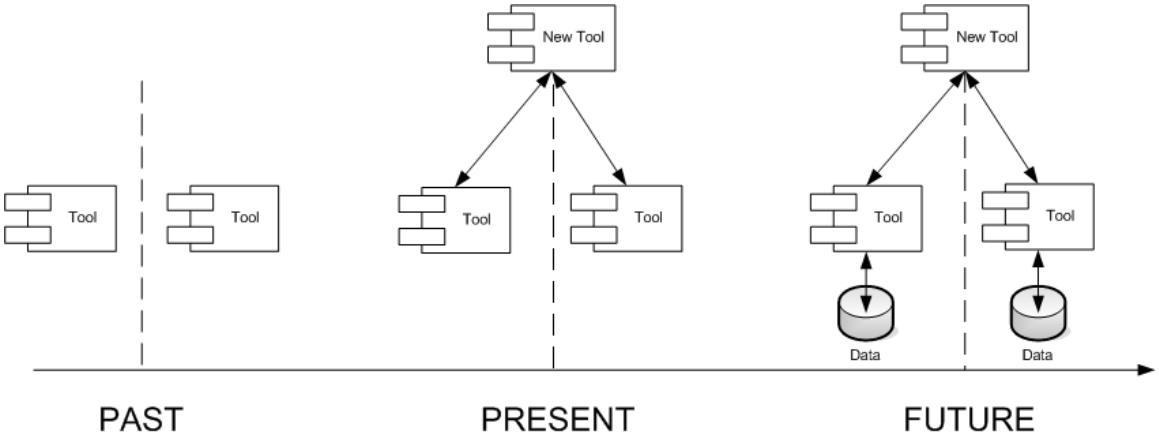


Figure 2 - Past, Present and Future Programming Environments

The above diagram is an adaptation from the work of Dr. Andreas Zeller, of the Saarland University Germany. Zeller (2007) believes that above all a programming environment must integrate into various other tools. Extending from this Zeller believes that further than just an amalgamation of tools, the tools must support automation between the integrated environments. These environments should be able to support one another. This synergy allows tools to be built upon existing environments, while exploiting the process and program data made available through this synergy.

Extending from this notion, this hypothesis proposes that such an environment can mirror concepts of found in popular social networking applications, such as folksonomies and presence management, and deploy them into a web based IDE. Through such an environment, software components can be tagged and labeled. The environment can be focused around semantic web and natural language, and notifications can occur in real time, which can be achieved over distributed environments.

Other concepts can be focused around being aware of someone's presence within the environment, including status updates, commenting, rating systems, and the ability to tag parts of code for review. Other people finding components can be integrated into the environment to find matches of people who fit a certain criteria or expertise.

If deployed as a complete platform, the environment can track development activities and provide statistical information for both the individual and the team as a whole. The development environment would be available anywhere any time, and offer complete transparency for the work being completed around you in real time.

There would be no need for equipment as the environment could simply be accessed over the Internet. Anytime you have access to the Internet, you have the power to access your complete development environment.

3.7. A MODEL PLATFORM

Given the need for real-time notifications, presence management and accessibility from any location, the desired platform is focused around cloud computing. A rationale for this choice will be outlined below.

Michael Miller, in his book, *Cloud Computing: Web-Based Applications That Change the Way You Work and Collaborate Online*, argues that a computing revolution is about to occur and that 'applications and documents are going to move from the desktop into the

cloud' (Miller 2008, p.2). This provides light on the emerging trend that is cloud computing.

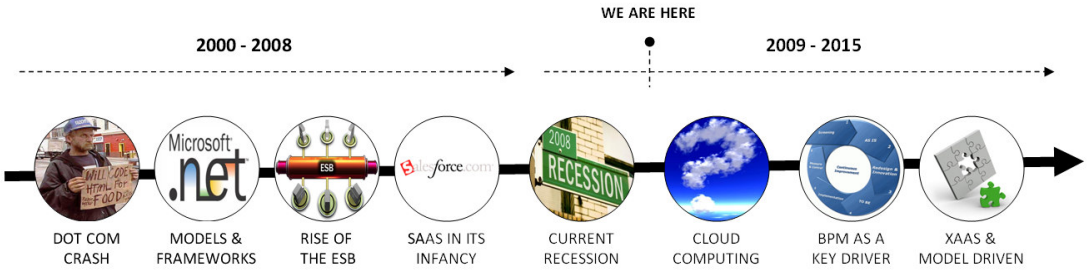


Figure 3 - Industry trends, 2000 through 2015

To focus more on this emerging trend, one can look at the predictions made by Woodward & Hallett (2008). The diagram above is an adaptation from these predictions. The model depicted indicates a number of transitions the industry has made since 2000. This diagram shows the fall of the dot com boom, which is followed by a number of models and frameworks that were established. This lead way to the rise of the enterprise service bus and SaaS in its infancy, such as in the example of Salesforce.com.

2008 through 2009 has seen a global recession, while the next big wave shows cloud computing at the forefront of where the industry is headed. Predictions also follow that beyond this will give rise to business process management, model driven services, and XaaS, where X is any arbitrary business requirement that can be maintained as a service.

This research aligns with the predictions made by Miller, Woodward and Hallett, and aims to establish models based around the next industry transition, while trying to do so in preparation for this innovation away from desktop oriented computing.

Cloud computing can not only be recognized as software as a service (SaaS), but it also becomes synonymous with platform specific ideals, such as platforms as a service (PaaS) and infrastructure as a service (IaaS). In fact there are a number of well established cloud computing platforms already in place, as list in the table below. The table has been taken from the work of Siebeck et al. (2009).

Provider	Amazon	Google	Sun	Salesforce
Product	Elastic Compute Cloud (EC2)	AppEngine	Grid Engine	Force.com
Website	http://aws.amazon.com/ec2	http://code.google.com/appengine	http://www.sun.com/software/sge	http://www.salesforce.com/platform
Style	IaaS	PaaS, IaaS	IaaS	PaaS, SaaS
Platform	Virtual Machines (Amazon Machine Images) running Linux or Windows	Python Web application containers or Java web container	Virtual Machines running Solaris	Apex (programming language of Force.com) Sandbox
Additional APIs	Amazon Web Services, e.g. Simple Storage Service (S3), Simple Queue Service (SQS)	Google APIs, e.g. Google Accounts (user management), Datastore API, Email services	-	Salesforce API, Connectors
Target applications	any	Web applications	any	Enterprise web applications

Table 1 - Comparison of existing cloud infrastructures

(Siebeck et al. 2009)

Current implementations focused on cloud computing can utilize a number of these existing services in order to take advantage of cloud architectures in the current market. A technical implementation behind the Google AppEngine will be discussed later in this report.

Although Miller's strong argument toward cloud computing, he highlights that cloud computing may not be the best architecture for everyone. He suggests that industry professionals assess whether cloud computing is right for them. Based on this notion, the following summarizes the suitability constraints Miller poses on cloud computing.

Suited for …	Not suited for …
• Collaborators • Road warriors • Cost-conscious users • Cost-conscious IT departments • Users with increasing needs • Users with fast changing environments	• The Internet-impaired • Offline workers • The security conscious • Anyone married to existing applications

Table 2 - Cloud computing, suited and not suited for ...

The main arguments behind Miller's predictions indicate that collaborators, road warriors, users with increasing needs, and users with fast changing environments are suited for cloud computing. These characteristics are strongly related to the requirements proposed by Herbsleb, Belloti and Bly, and de Souza. The increased need for collaboration aligns with the suitability for cloud computing collaborators. The road

warrior trait aligns with Belloti and Bly requirements for teams on the move. While it can be recognized that increasing needs and fast changing environments can be inherit in any software development practice.

Given the choice of platform, one must also recognize the advantages and disadvantages made by such a choice. The following is a list of advantages and disadvantages of cloud based architecture based on Miller (2008).

Advantages	Disadvantages
• Lower IT infrastructure costs • Increased computing power • Unlimited storage capacity • Improved compatibility between operating Systems • Easier group collaboration • Universal access to documents	• Requires a constant Internet connection • Doesn't work well with low-speed connections • Features might be limited • Stored data might not be secure • If the cloud loses your data, you're in trouble

Table 3 - Advantages and disadvantages of cloud computing

There are primarily a number of concerns surrounding security, and the reliability of data storage from cloud based architectures. Given the infancy of the platform, Miller also argues that features may be limited based on current availability. However, organizations looking to improve infrastructure in a dynamic and scalable manner can take advantage from the increased computing power, lower running costs and potentially unlimited storage. The key advantages evident are increased group collaboration and universal access to information. The advantages here outweigh the disadvantages; however organizations wishing to deploy such a solution should consider whether their corporation aligns with the cloud computing model.

Based on what has been presented concerning current industry trends, the availability of existing platforms, the high suitability for developers and tech savvy individuals, and the advantages outlined, a definitive rationale behind the cloud computing model can be seen.

3.8. WORKING TOWARD A SOLUTION

Whitehead (2007) draws attentions to the fact that several gaps exist in collaboration efforts in software engineering. He also draws light to the notion that desktop IDEs can enhance collaboration if they are integrated with web-based IDEs, and notes that such an investment has not been met within the software engineering community. It should be noted that this research agenda aims to fill a number of these gaps through research, framework implementation, and technical implementations.

Based on the challenges aforementioned, traditional software development has an inability to seamlessly communicate from one location to another. There are also a number of challenges surrounding distributed collaboration, and cross team transparency.

A solid foundation for framework has been seen through the investigations behind industry trends and the benefits of cloud computing. In fact Miller states that 'if you often collaborate with others on group projects you are ideal candidates for cloud computing' (Miller 2008). It is this rationale that will form the argument for the rest of the paper.

Through the notion of collaborative coding in the cloud, the following sections will assess the improvements in efficiency, communication and collaboration offered by this framework. This will be done by aligning with a number of social networking features, facilitated by cloud based architecture.

The rest of the report will focus on existing literature, industry benchmarks, a qualitative survey, the proposed framework, and a subset of research which was aimed at achieving a technical implementation behind cloud based communication and collaboration.

4. LITERATURE REVIEW AND RELATED WORK

There have been a number of significant research agenda and prototypes developed to in the areas of distributed collaboration, increased developer transparency and moving the IDE away from the desktop. The following section will review a discrete selection of these research dissertations. These include the Jazz framework developed by Cheng et al. (2003), the toolet system built for the JEdit IDE from Yap et al. (2005), a collaborative tool to increase developer peripheral awareness from Reeves and Zhu (2004), a collaborative web browser from Lei et al. (2004), and a project aimed at building an IDE to be used on a mobile device from Jarvensivu (2006).

4.1. THE JAZZ FRAMEWORK

The Jazz framework, coined to be synonymous with a musical Jazz band, is a research project from IBM that encompasses contextual collaboration by embedding a series of collaborative features into the Eclipse IDE. As previously discussed Cheng et al. (2003) describes contextual collaboration as 'an approach to collaboration in which users are not forced to leave their core applications to launch collaborative tools or visit a collaboration platform; instead, collaborative capabilities are simply available as components that extend standard applications' (Cheng et al. 2003, p. 21). They suggest that contextual collaboration has been at the forefront of many industry products, but has been largely overlooked by the research community. They also suggest that embedding collaboration seamlessly into applications saves users the time and effort associated with context switching to other tools when they need to communication with others in their development team.

Throughout their development efforts, Cheng et al. (2003) state that teams may utilize a number of general purpose collaborative tools that offer multi-user capabilities, such as chat utilities, online whiteboards or document repositories. Alternatively developers utilize any number of tools available to them, such a CAD tools or spreadsheets, and make use of a number of tools such as email and instant messaging to achieve

collaboration. In this case, artifacts end up among a number of email inboxes, file systems and other tools. Cheng et al. (2003) argue that chat and email are ineffective tools for achieving collaboration, illustrating the example that one might need to consider what medium a discussion took place, if and who saved a transcript and where it is currently saved.

The objective behind the Jazz project was to outfit core applications with contextual collaboration so users will know about other users actions, such as in the case when users are editing a file, debugging code or chatting with colleagues. The team can thus take advantage of this transparency among the team and increase design and communication efforts. This impact of team-level social interaction is aimed at increasing product quality and team performance. The team has an extra sense of peripheral awareness of the work, activities and discussions being conducted around them.

Jazz is designed to support small informal teams, where anyone can create teams and add or remove members from the team. There are six core added features to the IDE platform, as indicated in figure 4, which are discussed below.

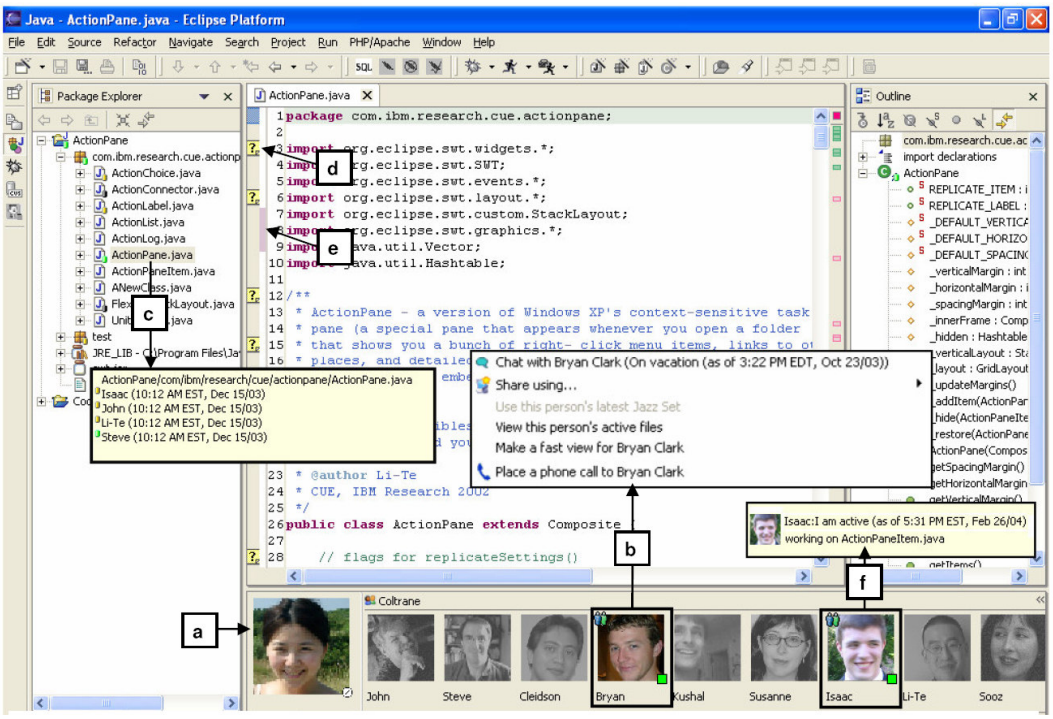


Figure 4 - The Jazz framework

The first of the denoted features (a) is the Jazz band, a shared buddy list which is a resizable panel at the footer of the interface. The view displays the teams the user belongs to, an avatar for each member of the team along with a coloured status icon indicating whether to user is online aware or busy, which updates in real-time. The second core feature (b) is a launching point of a variety of interactions, which is launched from the Jazz band. Right-clicking on a user will offer a menu to allow users to initiate a text chat, VOIP, or screen sharing sessions.

Feature (c) provides resource-awareness within the package explorer. Files in the explorer are decorated with colours to signify what colleagues are doing with the resource. Hovering over the resource brings up a tool-tip, if green the file is currently opened and being edited, yellow signals a file has been modified locally but not checked into the source control repository, and black denotes the file has been worked on but has been checked in.

Through feature (d), a developer can highlight a region of code, right-click and initiate a chat about it. When the discussion has finished a transcript can be saved and will be indicated with a marked in the left margin next to the relevant code. Team mates can later review the chat by clicking on the marker. Code can be annotated using the same technique.

Feature (e) is used to signal that a team member has modified an area of code, and hovering over the marker shows the difference between the local and remote code on a colleagues workstation. Feature (f) also allows one to display a tooltip that displays a user's status message, once hovered on the user's avatar.

Cheng et al. (2003) argue that contextual collaboration has the power to improve the working experience and the experience of working together. Jazz keeps users aware of who is working on what, who is communicating, where tasks are occurring and when they should be responsible for contacting others. Through the use of Eclipse plug-ins,

Cheng et al. (2003) has provided a benchmark for future collaborative IDEs and has done so in an innovative and efficient manner.

4.2. TOOLETS FOR THE JEDIT IDE

Yap et al. (2005) have implemented a proof-of-concept architecture to deploy dynamically and scalable extensions to the JEdit IDE through a series of discoverable Web Service modules. In this approach discrete software components are encapsulated into what they coin *toolets*, or more commonly refer to as plug-ins, which can be accessed as remote web service based components. The toolets are registered, discovered and dynamically integrate with the JEdit IDE once invoked.

Yap et al. maintain that there are common problems with current development environments, including the need to install tools on all user workstations and keep them up to date. They also promote the difficulty faced with tightly coupled PC IDEs and distributed server facilities, noting the complexities and incompatibilities between plug-in versions and remote software tools.

In this work, Yap et al. have extended the open source IDE to support new tool facilities via web service component registries. The components are invoked remotely via XML web service messages. A remote server is responsible for hosting the toolet services, and requests from one or multiple users are displayed within the JEdit client.

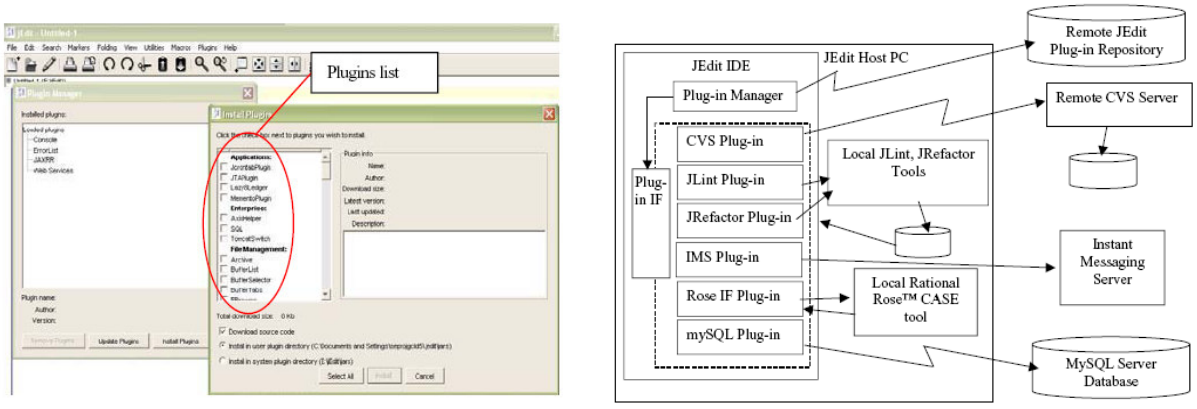


Figure 5 - Toolets for JEdit

To demonstrate the proof of concept Yap et al. has developed a number of toolets, including services for version control, collaborative message, code refactoring and code inspection. These examples were developed with third-party libraries, and web service wrappers to provide data and control integrations with the proposed architecture.

Yap et al. emphasize this architecture is best suited for collaboration and information exchanging, and may be of particular relevance for teams in need of collaboration, versioning, messaging tools, integration, greater transparency behind design decisions and test cases among the team.

Yap et al. highlight that IDEs provide developers with a means of integrating a variety of toolsets to improve personal productivity and code quality. They also argue that it is not only individuals who seek extensibility, but these integration strategies are also sought by software vendors who aim for increase productivity and quality from good IDEs platforms.

Whether through personal preference, or a culture dictated by the organization, the right selection of tool functionality can vastly improve development processes within an organization. Yap et al. goes further suggesting that software engineering tools seldom provide a complete set of functionality developers make use from. Thus software tool integration has become a major research and practical areas of software engineering, where developers can compose environments from multiple integrated tools.

Yap et al. consider web services as a powerful integration technology that can facilitate a variety of IDE extensions for software developers, and can be used for mandated tools within organizations or for dynamic requirements. This approach to dynamic and scalable IDE functionality allows users to build up a tool base as required, and provides an incremental approach to their integration requirements. Yap et al. recognize that web service architecture offers support for tool integration across data, control and process oriented facilities, any lack of distributed tool installation and upgrade process in place, monitoring and overall increase flexibility from the platform.

Yap et al. have provided an exemplar prototype which illustrates the power of web technologies, while also highlighting the importance of an extensible, dynamic and customizable IDE. They have provided much light in the way of internet based scalable IDE architectures.

4.3. MOOMBA – A COLLABORATIVE PLATFORM

Moomba is a collaborative environment which assists distributed teams in execution of the extreme programming methodology for global software development practices, developed by Reeves and Zhu (2004) of the University of Queensland. Moomba is an Aboriginal word meaning let's come together and have fun (Reeves and Zhu 2004, p.1). Reeves and Zhu recognize that global software development is an emerging trend which leads to communication overhead and separation among developers. They attribute this to advances in networking, telecommunication and internet technologies. Through such endeavors teams aim to jointly develop the same artifact in a collaborative manner.

Reeves and Zhu argue that due to this radical shift in the way software engineering is achieved there are many challenges surrounding these development practices. These challenges include project coordination, awareness, collaborative coding and effective communication, and are noted to primarily arise due to across site development and time zones.

Reeves and Zhu discuss the proven and popular Extreme Programming (XP) methodology, and its lightweight, user-centric and usability values, and the benefit this poses on software development. However, to exercise XP in a distributed fashion teams must introduced an increased level of communication and awareness to mirror the close-proximity values educated in traditional XP practices.

Moomba is designed to be a virtual workspace to support communication, awareness and a strong sense of coherency within a group. Reeves and Zhu pose that a collaborative environment should integrate with existing systems used by organizations, such as document repositories, version control, databases, video conferences and email systems. Through such integrations teams have access to improved communication, notification, document sharing and an increased aware of software development processes. Through

such workspace awareness, teams gain access to up to the minute knowledge about another group member's interaction within a shared workspace to collaborate effectively.

Moomba provides information on presence management along with the interaction team members have on the different software artifacts. Reeves and Zhu assert that through knowledge that a user is working on the same artifacts developers can resolve potential conflicts and any configuration problems that may arise. This knowledge also encourages team members to work together if they are both working on the same artifact.

Moomba has a collaborative editing system, where individuals have the ability to share parts of their documents with one another, while also having the ability to browser other areas of the project content without disrupting a colleague's position. This is aimed at allowing pair programming within a distributed manner. This feature also supports the collaborative coordination of XP story cards and other design documents. The editing system supports syntax highlighting, code completion and automatic indentation. Any changes to source code are propagated across to the respective team member's local copy.

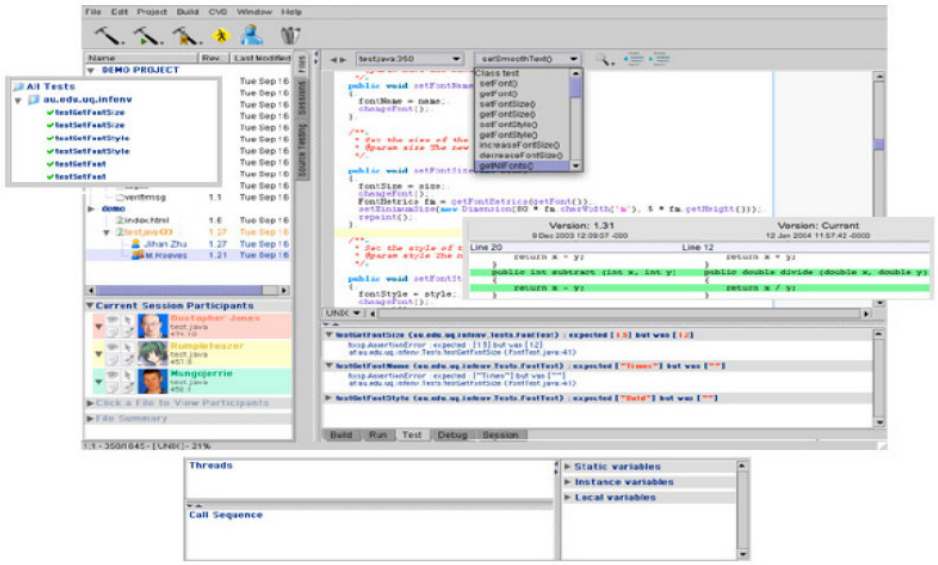


Figure 6 – The Moomba Environment

This collaboration is powered through three tiers, a web portal for project coordination, a remote collaborative server, and a collaborative programming environment to facilitate pair programming. These tiers are also integrated with version control systems and a user repository.

The web portal is responsible for managing indirect communication and has been customized to offer notifications and updates based on a team member's role. The system is also responsible for sending notifications to users who are offline, if events are deemed to be of significance to the user. Team members can also obtain information on the status of the current project and any associated release or iteration information.

Moomba also maintains a people finding component, where users can search for colleagues who match certain requirements or attributes they are interested in. These attributes are maintained through the user repository. This process is particularly useful for those individuals looking for others that meet pair programming compatibilities, a particular skill level or are interested in similar programming roles.

Moomba also supports collaborative debugging, where team members can collaboratively control program execution. Each member has access to information regarding the call sequence, each thread and the respective variables. This allows each team members to follow the execution of the program within the source code.

Reeves and Zhu propose that future implementations should aim to employ a match-making agent to replace the current people finding components. They also propose that video and audio conferencing facilities should be integrated into the collaborative editor. Once ready for public distribution they also state that the Moomba system will be released as open-source.

While the Jazz framework aims at providing intrinsic communication abilities with the development environment, Moomba has successfully implemented an alternate design that focuses on the core development processes to allow developers to collaborative in both design but also the underlying functionality and design elements of the project. This

can be recognized to greatly improve design, up-skill, cross-communication and overall team synergy.

4.4. COLLABORATIVE WEB BROWSER

Lei et al. (2004) have developed a collaborative platform to provide developers with contextual collaboration. Lei et al. draw on many of the same notions put forward by Cheng et al. (2003). Lei et al. discern that many tools used in typical corporate settings such as phone meetings, instant messaging and email have several draw backs. Such drawbacks include ad hoc support tasks and unstructured collaboration. Users have to explicitly switch and forth between these collaborations tools and business applications and move data between them in order to achieve an appropriate level of collaboration.

Lei et al. draws attention to the fact that industry analysts have predicted that one of the most critical management trends, is the shift toward embedding process-specific collaboration components within business systems. They also argue that contextual collaboration encourages solving problems as they arise and not as they grow.

The design rationale aims to achieve three main objects, those of integration, interoperation and extensibility. The platform should support easy integration of third-party collaboration to allow organizations to make use of pre-existing investments in this field. The platform should also enable interoperation among cross-vendor technologies found across inter-enterprise collaboration. The platform must also be easily extensible, allowing for an evolution into new collaborative capabilities.

Lei et al. emphasize that at any one time a user may only have access to a subset of collaboration tools, which may be dependent on location, time and current activity. This draws on the importance of a web based tool and internet technologies that can overcome many of these concerns.

Lei et al. denote that collaboration is essential to every software engineering domain. Further stating that members of a project team must be able to effectively plan, share, clarify, negotiate, brainstorm, coordinate and exchange information. They maintain that

this efficiency behind people to people collaboration has a direct impact on the quality of the final software deliverable produced.

The collaborative web browser produced is described as a systematic way to integrate collaboration features into web applications. The tool is designed to transform web browsing from an individual task to team based behaviour, enabling individuals to interact with a community and leverage collective knowledge and experiences through the browsing session.

The collaborative web browser is an extension to the Internet Explorer (IE) browser on the Windows platform. The original IE has been extended with a collaboration panel and collaboration explorer bar, which are integrated with a collaboration server hosted at a remote location. The collaboration explorer bar acts as a view that displays collaboration information specific to the current browsing context.

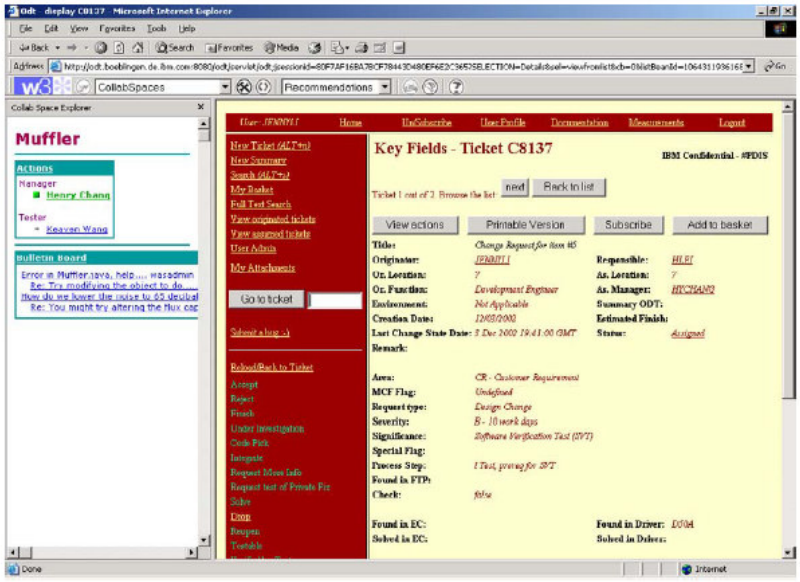


Figure 7 - Collaborative web browser

The remote server manages collaborative workspaces, and refreshing of the collaborative view for clients connected. Upon logging onto the platform users can receive collaboration requests from others. The collaboration panel receives events based on user actions in the main browsing panel. This information is then relayed back to the remote server for analysis, and associates a browsing context with the user. On review, users can initiate a shared browsing session with one another and share information.

While this implementation does not improve development practices, it provides a model for communication and collaboration in web based environments. In contrast to this research, this collaborative web browser was developed as a component within the browser, and not as a standalone web application. The research of Lei et al. provide an archetype for future models that wish to utilize web technologies with client-server architecture.

4.5. AN IDE FOR A MOBILE DEVICE

Järvensivu et al. (2006) delineate their experiences of creating an integrated development environment for a mobile device. Their goal was to deploy a complete IDE at the lowest possible costs. Through a number of open source communities and existing integration components they were able to achieve this goal.

The project coined Laika involved creating a user friendly integration development environment for software embedded within a Linux system. The chosen mobile device was the Nokia 770 Internet tablet (see figure 8). The device was recognized as a PDA rather then a mobile phone, and also runs the Linux operating system. The project aimed at integrating components from several open source development tools in order to seamless achieved a mobile based individual workstation.

The Eclipse platform was chosen to be the underlying framework used for the IDE. This was due to its open source nature and its vendor neutral open development platform. While there were alternatives discussed, Järvensivu et al. note that many developers are already familiar with the Eclipse platform, and the fact that its plug-in capabilities allow the platform to be extended and specialized for different purposes. Their decision to utilize existing open source components was elected to save time and technical resources. As such there were a number of third party open source modules that were implemented across the complete platform.

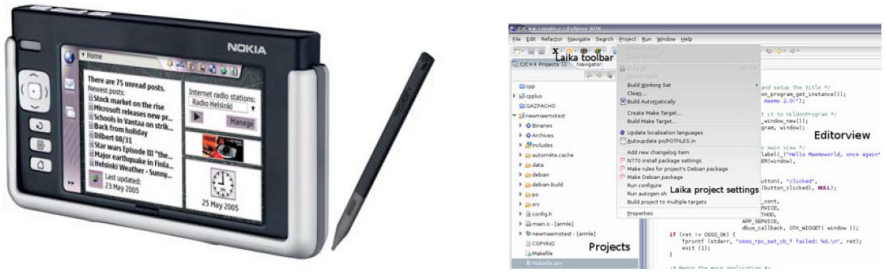


Figure 8 - IDE on a mobile device

The IDE was primarily designed for development with the C programming language. The development tool was able to utilize the underlying libraries from the operating system to build through make commands, compile through gcc commands, and debug using gdb commands. The tool was able to communicate with the external utilities and interpret their responses to test code. The tool also had intrinsic support for syntax highlighting, debugging, launching, parsing and searching. However, there were some limitations based the overall designed of the platform, as the small screen resolution meant that a number of features in the platform were removed, and there were visual concerns when testing software artifacts.

Interestingly, the Järvensivu et al. team outlines a lack of proper communication when developing the system, which let to a number of overlapping code modules. They attribute this to the lack of regular meetings and transparency among changes to the system that was being development.

A working model was released to a number of developers among development communities to test. These users were recognized as natural integrators of the system and were able to provide constant feedback on the design and implementation of the platform.

Based on their experiences with the project, Järvensivu et al. promote five major recommendations for those looking to implement similar platforms. The first is to have a clear goal in mind and define concrete short term goals to address the big picture incrementally. The second is to plan a strategy on how to integrate different subsystems. Third is focused on selecting appropriate partnerships through the particular industry or

research areas, and ensure you select cooperative and supportive partners. Following this, they advise to not get involved in too many development communities but rather take time to learn how the community operates, and whether your development habits align with this community. Finally, the implementation should prove to be something practical to allow designers to 'eat their own dog food' (Järvensivu et al. 2006, p. 57), and learn how the platform can be improved or any shortcomings that may be present.

Järvensivu et al. provide an empirical study on how researchers can deploy an IDE that is of a mobile nature and is not dependent on the desktop environment. Particular emphases can be placed on the recommendations made by the team for others looking to implement an IDE. This also provides a case where development projects often lack proper communication and project transparency, and it is only until developers can reflect on their experiences do they see the obstacles that are present during software development processes.

4.6. A NEED FOR FUTURE RESEARCH

Attention should be draw to the fact that while these research agendas and implementations have provided significant benchmarks and literature in field of communication, mobile based development and improved project transparency, these implementations alone to do address many of the research challenges addressed previously. This research aims to extend on many of the argument put forward by these past research areas but will also promote the use of cloud computing to better achieve such goals.

There are also a number of draw backs from these implementations. Such concerns are largely focused on the arguments that the implementations are still confined to the desktop environment, as seen with the Jazz platform. Another concern is the outdated use of technologies that may not be viable based on present-day technologies, such as the case for the collaborative web browser and its dependency on older versions of Internet Explorer. There are also implications around the reality that some of these platforms have

never been deployed within a production environment, and therefore lack the proper testing or user acceptance necessary in a comprehensive development environment.

The following sections will be aimed at addressing some of these concerns while also extending on many of the design principles and implementation recommendations put forward by Cheng et al., Yap et al., Reeves and Zhu, Lei et al., and Järvensivu et al. It should also be noted that while this list of research endeavors is not an exhaustive list in this field, the five discussed were raised based on their merit and strong correlation to the research presented throughout this paper. See Appendix for this correlation.

5. PIONEERING PLATFORMS

While the literature review outlines a number of past research efforts to address areas of collaboration, one must also look to the industry leaders in order to discover how the commercial arena aim to address similar problems. There are a number of tools available in this space, however this section will report on two major influential organizations, Google and Mozilla. The two major pioneering platforms in the area of online collaboration and coding in the cloud are Google Wave and Mozilla Bespin. The following will provide a brief overview of the products, and how they have provided benchmarks in their field.

5.1. GOOGLE WAVE

Google Wave is an innovative approach aimed at improving communication and collaboration through a number of web technologies. The architecture compromises of three areas, the product, the platform and the protocol. Online literature often use these terms interchangeably. The product layer is the web application, which allows people to engage in discussions. The product is an HTML 5 application developed with the Google Web Toolkit. The platform layer consists of a number of open APIs that allow developers to create their own robots and gadgets which can be embedded into the Wave. The platform also allows Waves to be integrated into a number of other web applications, similar to Google Maps architecture. The protocol layer is based on the storage and sharing of Waves, including concurrency control. The protocol is designed on an open federation, and Google has also released this protocol as open source. See figure 9 derived from Ferraté (2009).

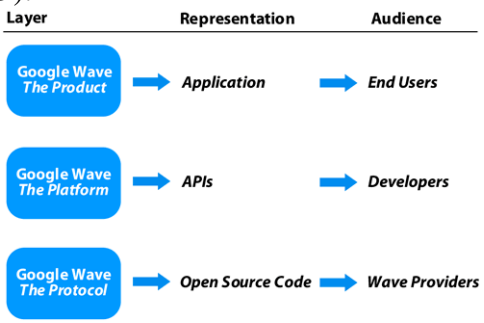


Figure 9 - Wave architectures

Wave is currently in BETA with limited user access, and was only released to the public September 2009. The Google Wave product is currently hosted by Google, however post BETA releases will allow organizations and teams to host their own Wave server, much like organizations host their own email servers.

Google Wave is recognized as a real-time collaboration platform that aims at providing the services of several web technologies including email, instant messaging, wikis, online documents and gadgets (Ferraté 2009). Dion Hinchcliffe, a popular technology evangelist notes that Google Wave "consists of a dynamic mix of conversation models and highly interactive document creation via the browser" (Hinchcliffe 2009). Wave utilizes a number of modern technologies such as HTML5 and AJAX, to allow users a range of rich features and functionality providing a similar experience to that of a desktop application. A Wave consists of several Wavelets, or threaded messages, which maintain zero or more blips, the basic unit of conversation within the Wave. Participants can either be human or task oriented robots. Figure 10 has been derived from Ferraté (2009).

Figure 10 - Google Wave

Wave maintains many of the features of email, such as message archiving, attachments, single or multiple recipients and rich text editing. The instant messaging features allow participants to exchange messages in real-time, view keystroke by keystroke, yet the threaded nature means that users do not have to be online at the same time to engage in conversation. Synonymous to wiki behaviour, a message can be edited or removed from a conversation regardless of the changes actually being made by the original creator of the message. Many of the document editing capabilities offered in Google Docs can also be

found in Wave (Ferraté 2009). An important feature is also its ability to playback messages in chronological order for users to gauge how the conversation has unfolded, or for those individuals who have joined the conversation at a later stage. Figure 11 is also derived from Ferraté (2009).

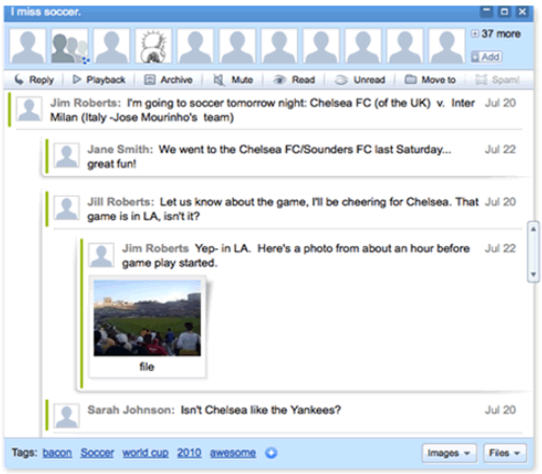


Figure 11 - A Google Wave thread

According to Andres Ferraté (2009) Google Wave will likely replace email and instant messaging, which are long since outdated. Ferraté even suggests that Wave may have the potential to become a de facto standard for communication and collaboration. Ferraté goes further in suggesting that due to many of its social networking aspects, there is potential for Wave to evolve and become a new type of social network tool to compete with platforms such a Facebook and MySpace.

Ferraté argues that people will likely turn to Google Wave because of its ease of use, speed, the real-time gratification, overall functionality, extensibility and its ability to integrate with third party applications and platforms.

Threaded conversations throughout Wave are highly interactive and provide users with content rich information. Information, gadgets and multimedia are all archived within the Wave, making the conversation highly accessible for anyone to access.

Wave allows users to view live messages as they occur, in part, allowing for faster communication without the need for idle time waiting for a response. These interactions, manipulation of extensions and concurrent editing occurs in real time. Utilizing the

Google Wave protocol, dynamic interaction between development teams and other stakeholders can be achieved (Joseph-Malherbe and Malherbe 2009). Albeit in BETA, Wave has provided a benchmark for online collaborative tools of the future, and has also demonstrated the true power of web applications available today. The Wave protocol, and its open format will allow a growth in the field of communication and collaboration available in the cloud.

5.2. MOZILLA BESPIN

Mozilla have developed a prototype cloud based code editing framework, coined Bespin, named after the cloud city located in the Star Wars movies (Broersma 2009). Mozilla have offered the online community a promise to provide a web-based code editor to increase developer productivity, enable compelling user experience and promote the use of open standards. Mozilla have promoted the strategy to make the platform open source, thus empowering developers to 'hack on the editor itself and make it their own' (Mozilla 2009).

Dion Almaer and Ben Galbraith, the founding developers of the platform, highlight that the goal of the project was to follow the example of tools such as Google Apps in shifting desktop-based tasks to the Internet (Broersma 2009). Almaer brings to light that "as a challenge, we wanted to take on an interesting project that you would normally think of as a desktop application, and see if it would fly on the Web" (Almaer 2009). Almaer argues that shifting an editor to the cloud should make it easier for developers to collaborate, and one goal of the project is to enable live-coding sessions (Broersma 2009).

The core functionality behind Bespin relies on the principles of ease of use, real-time collaboration, integrated command line, extensible and self hosted, fast, and accessibility from any location. Ease of user is focused on the editor experience, non-intimidating interfaces and should facilitate working with the code faster. The real-time collaboration is focused on share live coding sessions, and should easily accomplish collaboratively coding with one or more partners, although this is yet to be achieved in full. The command-line features of the tool are aimed at mirroring command-line tools from

popular editor tools such as vi and Emacs. The product is deemed to be highly extensible and offers and plug-in framework. Complex file processing logic ensures a smooth and responsive editing when working with large file sizes. The platform is also aimed at being accessible from any location or device using modern and standards compliant web browsers (Mozilla 2009).

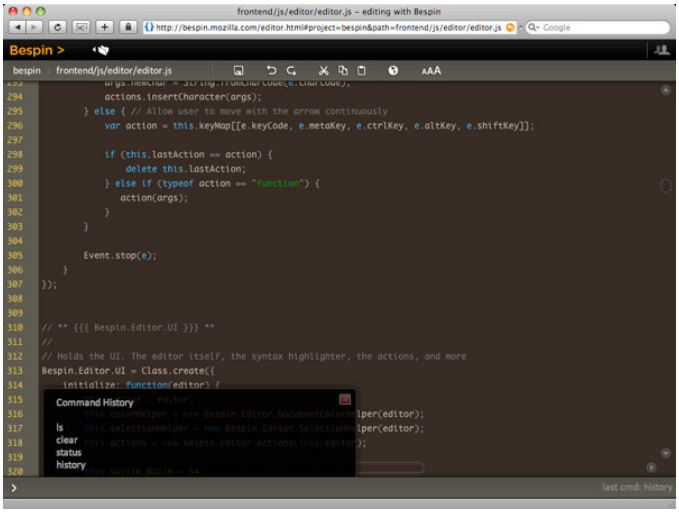


Figure 12 - Mozilla Bespin editor

The initial prototype offered support for editing, syntax highlighting, large file sizes, undo and redo commands, previewing of files, and an import and export feature. The product roadmap has also offered features around social coding, such as the ability to follow someone and allocate individuals to teams.

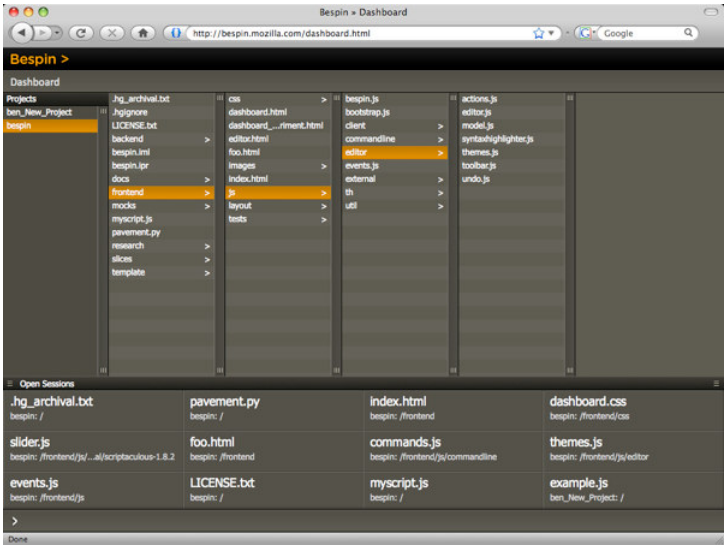


Figure 13 - Mozilla Bespin viewer

According to Paul (2009) Bespin is pushing the technology boundaries, and providing a vehicle for innovation. The creativity behind Bespin's architecture also demonstrates the power of web technologies available today. The open nature of Bespin has already proved a catalyst for innovation into this arena, with projects spawned which integrate Bespin with the Eclipse IDE, and another which integrates Bespin with XWiki (Paul 2009).

Bespin is also in BETA and was released mid-2009. The platform is available from Mozilla's website and requires a HTML5 compliant browser. Developers can also download a copy and host the platform on their own infrastructure. Bespin has a number of optimistic features, and has provided a benchmark for cloud based IDEs of the future.

5.3. PITFALLS WITH PRODUCTS

While many projects have aimed at utilizing Wave as an IDE, such as SameTimed an IDE architecture built on the Google Wave protocol (GoogleCode 2009), most endeavors lack proper development capabilities. Bespin also has promising collaboration features proposed in the product roadmap, although most of these are yet to be realized. The epitome of successfully collaborative coding in the cloud would be a merge of these two architectures, coupled with many features found with traditional development environments.

However, both platforms have provided an innovative and pioneering benchmark for such types of platforms in the future. Successful implementations of future collaborative cloud based IDEs should look to these tools and mirror successful concepts. Any such architecture which could employ concepts from both platforms would be at the forefront of this research.

6. THE NEEDS OF A DEVELOPER

Given many of the challenges discussed and a look into previous research, one must then study the needs of current software developers, and ask the question of what developers really want. It was determined the best method for a wide audience would be to conduct an online survey. The survey was aimed at collecting aggregate data regarding satisfactions and patterns of efficiency around software development teams, and their current Integrated Development Environment. The qualitative survey was constructed to engage industry professionals on their perceptions of collaboration, the types of tools they use to facilitate collaboration and their predictions on the future of cloud based development. The primary goal was to determine what IDEs developers use and why they use them.

During a one month period twenty-four questions across a number of topics were put forward to respondents. Answers were across a range of multiple choices, multi-selection and free response. To engage in a large audience the survey was completed anonymously, and each question was optional. Therefore answers need to be assessed on an answer by answer basis, as response numbers vary from question to question. Respondents were asked to take part in the survey regardless of their current roles, but rather have had development experience any time throughout their career.

The following outlines the intended target audience, the breakdown questions, procedure behind conducting the survey, data collection, the range of respondents that were received, key responses, and a summary of the results received.

6.1. TARGET AUDIENCE

The indented target audience was to engage a broad spectrum of industry professionals to reach a mass audience of respondents. It should be noted that the access to industry professionals was within a limited scope; therefore the majority of respondents were designed to be immediate colleagues or peers from university. It was determined that

respondents could be across any industry sector and experience level. The survey was also placed online to engage in a number of other individuals from differing countries.

6.2. PROCEDURE

A number of online survey tools were reviewed and prototyped for use, such as http://surveymonkey.com. However, based on the intended design of the questions, all such tools were deemed incapable of constructing the survey in the manner intended. Therefore a custom developed survey was built and placed on a personally hosted website located at http://thesis.domlovell.com. The survey was designed to allow for ease of use, have a number of different question types, and capture responses based on the answers of previous responses. Statistical data captured from Google Analytics reported that the average time a respondent spent completing the survey was 2 minutes. The survey was developed using a combination of PHP and jQuery, and was coupled with a MySQL database. There was over four weeks of design and development involved in developing the custom survey. The use of jQuery offered a number of event driven processes, such as asking respondents for a further detailed response based on the answer given on the multiple response questions. Respondents were also given the opportunity to provide comments on the survey, or contact the researcher directly. This allowed for a number of direct email communication from respondents who provided additional support or suggestions for the research agenda.

There were four main sections throughout the survey, general information, development and communication tools, satisfaction on current processes and survey submission. The first section collected non-indentifying information such as current industry, their role in the current position, education levels and years of experience in current role. This was intended to gauge the type of respondents that were completing the survey. The second section aimed at identifying development styles and general practices, questions were based on current IDE, development methodology, use of communication tools, team sizes and management styles that was practiced within their organization. The third section was based on developer satisfaction and offered a number of Likert scale questions, where the individual rated their opinion of a statement on a scale between Strongly Disagree and Strongly Agree. Most questions were focused on team management or their

ability to collaborate. There were also eight follow up questions throughout the survey, and respondents were asked to provide a textual responses based on their previous selection. This offered a number of extended responses which will be assessed in the following sections. The fourth area offered a survey confirmation, and also gave respondents the opportunity to provide their email addresses for those wishing to be sent a copy of the results. Questions were ordered from general to most specific, with the final question asking respondents their perceptions on an IDE hosted in the cloud, and any barriers that may prevent this from occurring. For a full list of questions, refer to the Appendix.

6.3. PARTICIPANTS

The survey was advertised through a number of collaboration mediums, such a Twitter and Facebook. Using a number of predefined Twitter messages and hash tags, the survey was advertised at set intervals throughout the day. These messages were also propagated across to Facebook. Several individuals were also able to convey these messages on their own personal Twitter accounts, including Ben Galbraith and Dion Almaer the founding individuals behind the Mozilla Bespin project. This process was able to engage in an immediate range of responses. Following this, the survey was advertised throughout the Google Wave and Mozilla Bespin Google Groups forums, which members are notified of messages via email and through the online forum. The survey was also advertised throughout a small range of university students at the University of Technology Sydney. Further advertisements were completed via word of mouth from industry colleagues.

Medium	Visits	Percentage of traffic
Direct traffic	77	44.51%
groups.google.com	37	21.39%
twitter.com	26	15.03%
facebook.com	17	9.83%
mail.google.com	9	5.20%

Table 4 - Survey traffic

The table above indicates the statistical data from Google Analytics. Most traffic was received via popular collaboration mediums such as Google Groups and Twitter. This gives an indication on how respondents were exposed to the survey and the communication mediums respondents used. Over the one month period there were 173 visits to the survey, of which *77 responses* were collected in total from 14 different countries. The majority of responses where received from Australia, the United States and Malaysia.

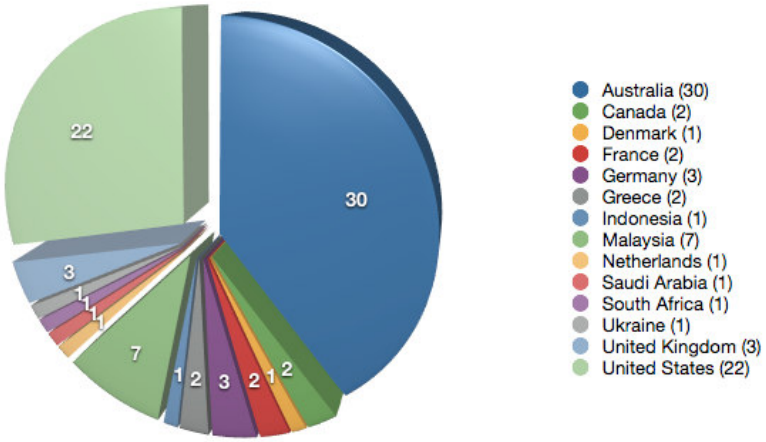


Figure 14 - Question 1: Country

6.4. DATA COLLECTION

After the one month period the survey was closed for responses. Data was then extracted from the database and aggregated. A number of responses offered an *other* or *None of the above* option for which respondents could provide free-form responses. These responses were analyzed and aggregated into similar categories. The Appendix outlines the complete data set in both tabular and graphical format, along with the complete set of free-form questions. An appropriate sub-set of the data will be assessed below.

6.5. AGGREGATE DATA REGARDING RESPONDENTS

The spread of industry professionals were vast, with the majority of individuals working in high areas of technology or for independent software vendors. The major job roles were found to be developers, engineers, students and consultants. This indicates that most responses are received from individuals that are currently in a development role.

The majority of respondents operate in a team of 2-5 people, with the second largest portion of respondents operating in a team between 5-10 team members. From those respondents that answered the question, it was found that most respondents indicated their teams operating under a distributed environment with individuals across multiple sites.

The top three development methodologies were found to be Agile development, followed by Extreme Programming, followed by a Waterfall method. This pattern was mirrored throughout the management styles, with respondents indicating that empowered workers was most common, followed by a democratic style, and the third most common being paternal or semi-dictatorial styles. This becomes relevant for discussions throughout later sections of this document.

6.6. KEY RESPONSES

Respondents were asked what IDE they used from a list of common IDEs, or could indicate their own IDE used if not listed. Upon a response, an additional question appeared asking for an extended response as to why they used this IDE was offered. The most common choice of IDE with almost half of respondents indicating their choice of IDE was Eclipse. This was followed by Visual Studio and IntelliJ as the second and third most common IDEs. Interestingly, despite active advertisement from the Mozilla Bespin team, only three respondents indicated they actively use Mozilla Bespin. This provides a small window into the adoption patterns of this tool, which will be discussed in later sections.

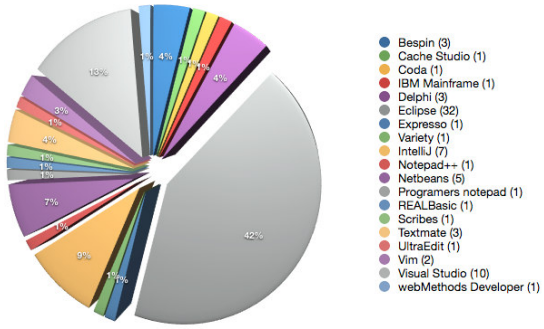


Figure 15 - Question 10: The majority of my software development is completed using the following Integrated Development Environment (IDE)

Overall responses on the choice of IDE varied, though one can look at the key responses to gauge the strategic decisions developers have made. Three key responses on choice of IDE can be focused on Eclipse, IntelliJ and Bespin. One response noted that Eclipse was 'mature, solid, good selection of related plugins, intuitive', while IntelliJ had 'great refactoring, not a lot of bugs (relatively speaking), easy to navigate and use, good code completion and other handy time-saving tools'. Bespin, the cloud based IDE was the choice of one respondent 'because the ability to access it from any web-enabled terminal in the world has changed my entire outlook on keeping downtime to a minimum.'

Respondents were also posed the question of whether their IDE met their expectations on the needs of a developer, and were also asked what they would change about their current IDE given the chance. Sixty percent of respondents indicated that their current IDE does what they expect of it. One respondent indicated that Eclipse met their expectations, yet improvements could be made around 'better integration with git, increased speed, better communications support'. Anther respondent indicated that IntelliJ met their expectations, however there lacks an appropriate level of performance from the tool, as they are using '2 quad cores + 9GB of RAM + a SSD to get it humming.' In contrast to these responses, one respondent noted that the Textmate tool did not meet their expectations and the tool required 'better collaboration support... easier process for configuration'. Based on these responses, there is affirmation that the principle needs surrounding an IDE are those concerning communication, performance, extensibility and collaboration.

The larger part of Likert questions were received with an agreed response, such as the notion that pair-programming aids development processes. 43% of respondents agreed with this idea. 48% of respondents agreed their team manages communication effectively, and 47% of respondents agreed that knowledge and expertise is shared among the team. 44% of respondents agreed they are aware of what others in the team are doing, and 45% of respondents agree they understand their colleagues design decisions. Over half of respondents indicated that social networking and collaborative tools aid development efforts during project delivery, with 55% agreeing and 20% strongly

agreeing with this notion. This is largely a result of the chosen development methodologies, surrounding Agile and Extreme Programming methodologies.

The most common collaboration tool used by respondents was instant messenger, followed by issue tracking software, wikis, followed by forums and blogs. This indicates that almost all respondents rely on instant real-time communication during development process, but also primarily rely on knowledge sharing or knowledge base tools during development.

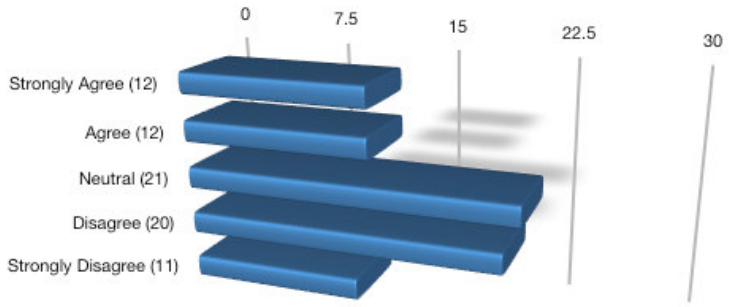


Figure 16 - Question 24: I would prefer to have my IDE hosted in the cloud, rather then on my workstation

The final question that was posed to respondents was whether they would prefer to have their IDE hosted in the cloud (Figure 16). As indicated from the graph above, a mixed range of responses were received. However, the 76 responses to this question are negatively skewed. This would indicate that these respondents are not confident that an IDE can be hosted in the cloud. We can look at the extended responses to discover the reason for this disapproval.

One respondent offered a neutral response, and provided the extended response that ‘it is likely that you'll see cloud-based IDEs within the next few years’, yet also noted there is already a large investment into is desktop based environment, and these tools would need to be rewritten ‘which is going to take a lot of time and effort.’ Another respondents indicates they strongly agree with the notion of cloud based IDEs, and suggests that ‘cloud computing will strengthen our ability to perform quality assurance of the code product, it will also allow enough knowledge to be shared so the maintenance cost of the

customized code goes down, and (for public sector) provides a Total Lifecycle view of the enterprise with which this product will be integrated.' In spite of this, the respondent highlights the security implications surrounding cloud architecture, noting that 'fear of something catastrophic hitting the cloud and thus forcing the Government to shut it down for a spell-- is worrisome.' Alternate to this view, another respondent strongly disagrees with this concept, and explains they may be 'old fashioned or maybe my internet connection isn't fast enough', and goes further to argue that their current workstation has performance issues and their 'workstation IDE lags occasionally already', and remarks that any successful cloud based IDE 'requires fast response to keep up with me on the cloud.'

6.7. SUMMARY OF RESULTS

A review of the aggregate data illustrates that most respondents believe that communication and transparency is maintained throughout their teams. This can be considered a result of the development methodologies. The Eclipse IDE was the most popular choice of IDE, while respondents were primarily concerned with communication and performance from their IDE. There were mixed responses of the concept of a cloud based IDE, but ultimately this was related to concerns of security and performance from cloud architectures. Through a small range of developers across a number of countries world wide, the survey has provided answers to developer predictions on the future of cloud based development, while also determining what IDEs developers use and why they use them.

7. PROPOSED FRAMEWORK

The following section outlines the proposed framework, which will be completed through a requirements checklist, target audience, a list of potential barriers, and key areas for success. While this research does not implement any concrete model of a cloud based collaborative development environment, it is aimed that through this study future researchers and industry professionals can source this information to become the model for their own implementation.

The internet has demonstrated that social-networking can be useful for online communities and facilitates context awareness among individuals. A number of existing web applications such as Twitter and Facebook provide real-time ubiquitous communication and social interaction. It can be recognized that many of these platforms are focused on the individual and lifestyle affairs. Although, Cohen and Clemens (2005) outline that there are a number of professional focused services such as MSN Spaces or LinkedIn, the models are focused primarily on the individual and not the organization or a team.

Koskinen (2006) poses the questions of whether web 2.0 and 'social computing have anything to offer industrial business environments' and 'what about business-to-business usage?' (Koskinen 2006, p. 381). Koskinen describes these social web applications as an enabler of people, allowing them to collaborate through computer mediated communication. He also predicts that there is a lack of integration between existing systems and emerging social software, and that the future relies on software vendors to develop and integrate social software functionality into the next generation of tools and systems.

It is this school of thought the following requirements will follow. Albeit this framework will not draw reference from existing web applications, a generic set of guidelines will be outlined for others to follow. This is primarily because the framework is focused around an implementation in the future, and the framework should not be tightly coupled with

current implementations such as Facebook or Twitter. In such a case, the dynamic and fast changing environment surrounding Internet technologies may find many of the specifics behind such implementations obsolete in the immediate future.

7.1. REQUIREMENTS

There are three main streams researchers should follow when conducting implementation processes, those focused on a SaaS based solution, folksonomies and presence management, and collaboration capabilities. These requirements have been derived from the hypothesis produced in earlier sections, and extend areas of the research previously discussed. The checklists provide a series of possible features that can be implemented throughout the intended model.

Based on the requirements listed in previous sections, a dynamic and scalable checklist can be outlined. This should align with the concepts discussed in the model developed by Yap et al. (2005). The platform should offer ease of accessibility and maintenance, and follow the no download and installation ideals.

SAAS SOLUTION
• No downloads should be required from the end user • No installations should be required from the end user • Maintenance should be performed on the server side, and the application should not require end users to upgrade the systems • The platform should not dictate where and when tasks can be completed • The platform should track development activities and provide aggregate data for teams to analysis their efficiency levels • The platform should be available anyway, anytime with limited downtime and maintenance, which will be reflected by the choice of the cloud infrastructure • The ultimate goal should aim to provide developers with a complete development environment anywhere they have access to the Internet

Table 5 - SaaS checklist

Once platform and infrastructure models have been efficiently setup, researchers can then focus on the discrete functionality offered by the platform. Inherent in any true IDE the platform should offer the fundamental capabilities such as syntax highlighting, auto completion and pre-emptive error tracking. These will not be discussed in detail here, however researchers should be aware that without such features the platform cannot be deemed an appropriate integrated development environment.

The main areas surrounding communication should be facilitated through folksonomies and presence management. Vander Wal defines folksonomies as 'the result of personal free tagging of information and objects (anything with a URL) for one's own retrieval. The tagging is done in a social environment (usually shared and open to others). Folksonomy is created from the act of tagging by the person consuming the information' (Vander Wal 2007). These folksonomies should facilitate seamless and embedded communication methods which can support access to technical knowledge, design decisions and code reviews. Presence management will allow developers to be aware of the components of the system others are working on and their changes as discussed from the requirements of Bellotti and Bly (1996).

FOLKSONOMIES AND PRESENCE MANAGEMENT
• The platform should offer complete transparency on an individual's activities where appropriate • Those activities not appropriate should not be broadcasted to team members • Software artifacts and source code should have the ability to be tagged or labeled with meaningful contexts • The platform should provide a mechanism for real-time chat • Notifications and event driven messages should be broadcast to individuals in real time • The platform should offer presence management for each individual, including online, offline and busy status.

Table 6 - Folksonomies and present management checklist

FOLKSONOMIES AND PRESENCE MANAGEMENT CONTINUED …
• Status updates and real-time feed on activities should be publishable on the platform • Code reviews should offer a rating system. This can potentially be coupled with functionality roadmaps and release processes. • Areas of code should be taggable for review, or brought into another area to be approved or rejected.

Table 7 - Folksonomies and present management checklist continued

Following implementation of folksonomies and presence management, researchers can fulfill the collaboration requirements of the platform. This aims at providing cross site development capabilities, promotes pair-programming, cross-skilling and provides increased transparency among the development teams.

COLLABORATION CHECKLIST
• The platform should offer a people finding component that aims to source individuals who match a certain criteria or expertise • Code, artifacts and code reviews should offer developers a threaded discussion or comment area. • The platform should allow two or more developers to work on a piece of code at the same time and display changes on the code at both sides, providing developers an opportunity for cross-skill and pair-programming • Where appropriate, two or more individuals should be able to debug code and follow program execution. This may be relevant for JavaScript development, given the web architecture.

Table 8 - Collaboration Checklist

These three areas will provide the key to successfully implementing a collaborative coding environment in the cloud. These requirements should not be solely developed from ground up, but rather through a combination of new facilities as well as integration into existing platforms. Such a task follows the model put forward by Zeller (2007), as depicted from figure 2.

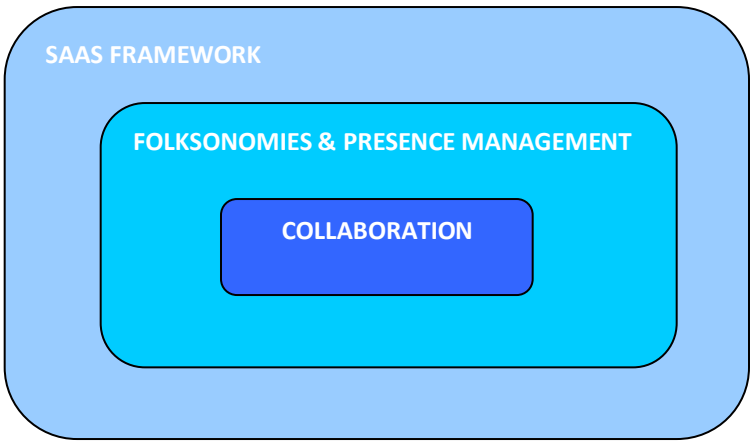


Figure 17 - SaaS, folksonomies and presence management, and collaboration features. Three core areas of the platform that are interdependent.

7.2. TARGET AUDIENCE

Section 3 highlighted the fact that cloud computing may not be appropriate for all individuals. Similarly it should be noted that this platform is not intended for all development teams. The intended platform is a purpose built architecture that aims at improvement communication, pair-programming, code reviews, small iterations and fast changing requirements, as described in previous sections.

Given the fast paced nature of the Internet and its changing demands, an Internet based IDE would be best suited for teams that follow an Extreme Programming, Scrum or other Agile methodology which supports fast changing requirements, team transparency and pair-programming. Organizations which are fast moving and non-autocratic can embrace the intended changes (Beck & Andres 2004). Other development methodologies and management styles may be challenged by an increased demand for change and a fast moving environment. However, the platform is not limited to those in an Agile methodology, in fact those individuals or teams who are not happy with their existing environment and have a willingness to change can also benefit from the target platform.

7.3. BARRIERS

There are a number of barriers and challenges that are applicable for this environment, and researchers must be made aware of these factors. These challenges have been derived from some of the challenges highlighted by Miller (2008) as well as a number of sound arguments received from the qualitative study.

These challenges include browser incompatibilities, network latency, adoption and willingness to change, cross compatibility with other development tools, security concerns with cloud computing, and the ultimate success or failure behind the cloud computing model. Researchers should address some of these concerns throughout their implementation.

7.4. KEY FACTORS FOR SUCCESS

As outlined by Cheng et al. (2003) the features of the environment must seamlessly be integrated into development practices. This ensures that habitual development routines are not hindered by the environment, but rather provide embedded support. Given this, a successful platform must not hinder development practices.

As seen in the figure 2 and emphasized by Yap et al. (2005) and Zeller (2007) a successful model must integrate with other software development tools. It is recommended that researchers aim to provide syndication or direct integration with task oriented development processes, where bugs and features can be tracked, to allow developers to monitor activity streams and identify the relevant tasks at hand. This implementation can follow model from the Mylyn Eclipse plugin which supports bug tracking and task orientation within the IDE (Eclipse Foundation 2009). Following this objective and based on the target audience, implementation can also focus on offering an integrated model into continuous integration systems. According to Grossman et al. (2004), true development transparency cannot be achieved without proper testing and continuous integration. Internet accessible continuous integration servers, such as

Atlassian Bamboo (Atlassian 2009), and the respective results can be syndicated or broadcast into the environment through areas of the checklist items proposed.

The proposed people finding component can be made possible through tool such as the Sonar system (Trampoline Systems 2009) which provides support for best path connections between individuals in an organization, or those individuals in the organizations who are at the central hub of communication between parties. A further suggestion, is time invested in an LDAP interface that allows the environment to have access to individuals within the organization and gain access to their extension details and position with the organization. Such application would be relevant to large organizations with multiple teams that are utilizing the environment.

These are suggested practical examples that would facilitate the need for integration with third party applications within the environment. Through such application increased adoption may be realized and provide the potential for success.

7.5. SUMMARY OF FRAMEWORK

The framework proposed highlights the three major areas of implementation, that is, SaaS based architecture, folksonomies and presence management and embedded collaboration features. A checklist of intended goals has been outlined, as derived from this research and hypothesis. It was deemed that the best suited audience for the platform was those teams who follow a fast changing, tightly integrated methodology such as those in the Agile methodology. A number of barriers were outlined briefly to ensure future researchers are aware of potential pitfalls. While a number of practical integration modules were suggested to align with successful adoption.

Though no technical implementation of the platform is achieved throughout this research, this proposed framework aims to provide benchmark criterion for individuals seeking to implement this type of framework in the future.

8. FACILITATORS AND OBSTRUCTERS

There are a number of emerging trends, adoption patterns and drawbacks behind cloud computing. The following section outlines a number of factors evident that may provide an increased industry push, successful adoption of the proposed framework, or the rapid decline of cloud based architectures, and are a range of elements future researchers should be responsive toward.

8.1. EMERGING TECHNOLOGIES

There are a number of emerging technologies that give insight into the nature of cloud based applications, and illustrates the contributing factors toward the success of this architecture. The following section will bring to light the benefits provided by the Chrome OS, Mozilla Prism and Google Gears, and their contribution to successful cloud based architecture.

8.1.1. CHROME OS

Chrome OS is a project from Google which aims to redefine user experiences by tightly coupling an operating system (OS) with features traditionally found in a browser. The entire experience takes place within the browser based OS and there are no conventional desktop applications. As such, users do not have to be concerned with installing, managing and updating applications. All applications will be launched within the browser and hosted in the cloud from their respective services, such as email, video and other popular web based applications. Google has released a statement on their blog stating the OS will be available later 2010 (Sengupta and Papakipos 2009).

According to Sengupta and Papakipos (2009) this architecture provides significant security benefits, as the operating system doesn't trust third party applications and will maintain them within a security sandbox making it harder for malware and viruses to infect the computer. Sengupta and Papakipos also highlight the performance improvements of the OS, as many of the bloated applications that reside on a normal desktop are not present and users 'can go from turning on the computer to surfing the

web in a few seconds' (Sengupta and Papakipos 2009). Google has released the code base as an open source project to engage with partners and the open source community.

The OS will be focused around established web applications with large user bases, such as Gmail, Google Docs, Facebook or Twitter. It will not be shipped with any conventional desktop applications. These web applications will act like native desktop applications, but will also be backed by local resources available on the desktop, such as offline storage or location information. The user experience is designed to be synonymous with the experience in the current Chrome browser. Users will not need be concerned with installing or maintaining binary distributions for the underlying OS. Although, the OS will be tightly coupled with a number of Google approved devices, and users will not be able to use it to replace operating systems on current desktops (Herrman 2009).

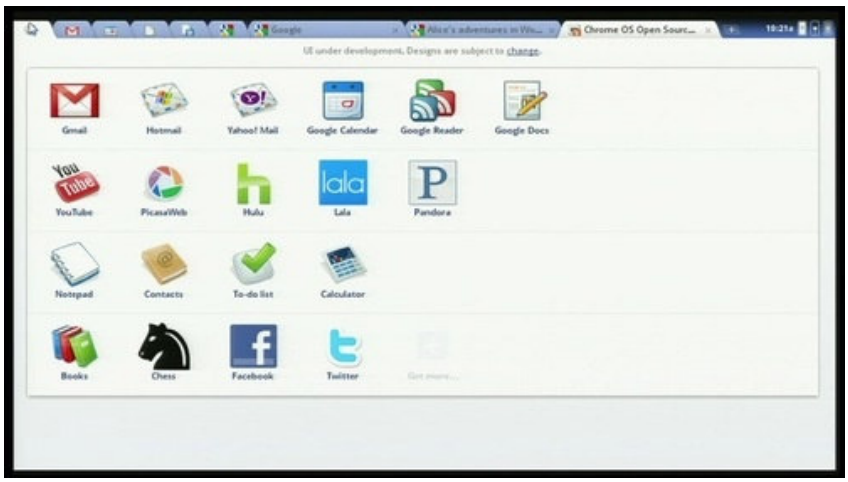


Figure 18 - Chrome OS

The core of the functionality relies on browser tabs and browser like panels. Your favourite web applications can be listed on a unified tab (see figure 17). Other small windows can be launched to provide access to chat and music players which can sit above your active tab, or can be docked to the left or right of the tab.

Herrman (2009) asserts that Chrome OS is taking the operating system totally online and 'the browser is already at the centre of most people's computing experience' (Herrman 2009).

This innovation from Google provides another opportunity for researchers to see the potential of the web and cloud based architecture available in the coming years. Chrome OS is an exemplary view on how the blur between desktop and browser is a common outlook for users in the near future, and strengthens the belief that applications can be based in the cloud yet can echo the features traditionally associated with desktop applications.

8.1.2. MOZILLA PRISM AND FLUID

Mozilla has released a product coined Prism, which acts to integrate single site browser instances into the desktop. Users can "split web applications out of the browser and run them directly on the desktop" (Mozilla Labs 2007). The tool allows users to create dedicated applications for their most popular web applications, associate icons with them and run them as standalone applications on the desktop. Once launched, a dedicated browser window launches the web application and maintains its own profile and preferences. The standalone instance also maintains its own system processes (Connaghan 2008).

Figure 19 - Mozilla Prism

Mozilla argues that personal computing is in a state of change, and more users are interacting with web based applications rather then desktop applications. However, this often means that the dedicated views toward these applications are hindered by the document-centric interface associated with web browsing, back or forward buttons, along with location bars are becoming obsolete when accessing these finite web applications.

Mozilla pose they are attempting to bridge experience between desktop and web based applications. Mozilla draw contrast to Adobe AIR and Microsoft Silverlight, stating that rather then building proprietary platforms for the web, they aim to 'identify and facilitate the development of enhancements that bring the advantages of desktop apps to the web platform' (Mozilla Labs 2007).

Connaghan (2008) notes that many IT managers are installing links to cloud based services on their user's desktops, yet tools like Prism are making this process an even easier task to achieve. Prism allows many web based applications to look like native applications on the desktop, without users needing to be aware of the underling cloud based architecture.

Mozilla aims to integrate popular web applications with the desktop by allowing them to be launched independent from the default browser. Users can interact with these applications in both online and offline modes, thus giving the applications a connectionless feel (Mozilla Labs 2007). These site specific browsers do not have menus, toolbars or features found in traditional web browsers, but rather focus on providing a discrete focus on the functionality and purpose of the web application itself.

Inspired by Prism, Todd Ditchendorf a former developer at Apple has created the Mac equivalent based on Safari's WebKit rendering engine. Ditchendorf (2007) states that any standalone applications created by Fluid are native Cocoa OS X applications offering seamless integration into the Mac OS.

This is exemplified by its abilities to integrate with OS X's cover flow abilities, offering previews of web documents or sites listed on popular social networking sites (see figure 20 from Ditchendorf 2007). Any web application can be offered as a standalone application and can be placed on a users dock (see figure 21 from Ditchendorf 2007).

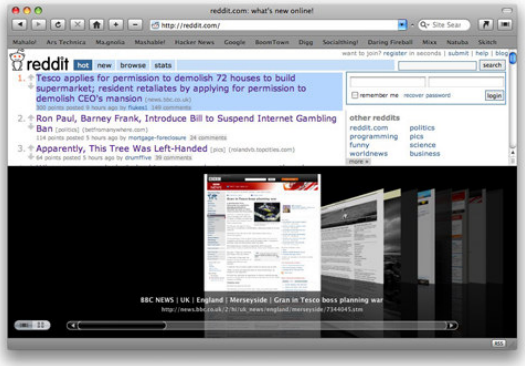


Figure 20 - Fluid Coverflow

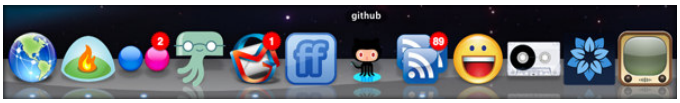


Figure 21 - Fluid docked items

Fluid also offers a number of enhanced user experiences by offering built-in user scripting, ULR pattern matching for browsing whitelist and black list sites, bookmarks, custom icons and additional features which integrate with the OS X dock.

These tools provide innovative and leading architectures which show how seamless integration between desktop and web based applications can be achieved. Powered by cloud based architecture these single site browsers allow users to utilize the power of web based applications without being hindered by some of the pitfalls of desktop infrastructure as described in earlier sections.

8.1.3. GOOGLE GEARS

Google Gears is a tool used to offer improvements to web applications, through natural interactions with the desktop, providing the ability to store data locally in fully-searchable databases, and offer improvements to JavaScript execution and performance. Chief Executive Officer of Google Eric Schmidt states that "with Google Gears we're tackling a key limitation of the browser in order to make it a stronger platform for deploying all types of applications and enabling a better user experience in the cloud" (Schwartz 2007).

The open source project is aimed at offering new features to web browsers. The LocalServer Cache can serve HTML, JavaScript and images locally, thus offering online

and offline modes of web applications. The database offers a fully-searchable relational data store available through the browser. The WorkerPool allows JavaScript operations to run in parallel, thus increasing performance and lowering resource intensive operations behind JavaScript operations. Traditional JavaScript execution temporarily halts operations until the event has completed or the script has loaded.

Reimer (2007) states that Google Gears 'blurs the line between online and offline applications by allowing users to continue working even without an Internet connection'. He goes further to suggest that if Google can quickly integrate these features with the online Google Docs and Google Spreadsheets tools, it could offer a viable future toward web-based applications and an increased adoption of these cloud based tools.

Google Gears not only provides a number of increased performance abilities to web browsers, but also draws attention to how cloud based applications can provide the necessity for both online and offline content behind cloud based applications.

8.1.4. SUMMARY OF EMERGING TECHNOLGIES

Chrome OS, Mozilla Prism, and Google Gears all exemplify how emerging technologies can facilitate cloud based architecture for the future, but also illustrates how such technology can provide performance enhancements and greater user experiences to web based applications. Thus emphasizing the trends toward cloud based architectures in the current market. Researchers should be aware of these impacts for future implementations.

8.2. OPEN SOURCE AND THE NEED FOR EXTENSIBILTY

A resounding argument throughout this research has been the importance of providing products to the open-source community, and the importance of maintaining an extensible framework. This was echoed throughout the emerging technologies, and was a common argument for the choice of IDE given in the survey conducted.

According to Bruce et al. (2006) open source software provides a number of benefits over commercial or proprietary software, such as a stronger customer involvement, lower costs and better quality. Any successful implementation of a collaborative coding

platform hosted in the cloud should look to promote open source. Stewart et al. (2005) has found that the more available a product, the more likely it is to be embraced by the development community. This process opens doors for developers to customize the solution to their own requirements and allows organizations to see the benefit of deploying such tools to meet their internal needs. This openness often promotes growth, and through such growth adoption and popularity of the product increases.

Future IDEs will need to examine successful IDEs of the present, such as the Eclipse framework. Gamma and Beck (2003) denote the importance of plug-ins and the significance that an extensible framework offers platforms, noting one of the key success factors behind the Eclipse framework is its ability to shape and model the tool based on a number of available plug-ins. The success of a plug-in framework is also tightly coupled with the openness and transparency behind the platform, as the more developers have access to the underlying framework, the more they can tailor and adapt the framework to their needs (Clayberg and Rubel 2009). Eclipse's open source nature has ensured that the product has been widely adopted, with tens of thousands of downloads of the product each week (Burke 2004).

This resonates with Google Wave's pioneering platform, and is epitomized by Google's Vice President of Engineering Vic Gudotra in the 2009 Google I/O conference on Wave. Gudotra states that Google 'need developers to help complete this product' (Google 2009). This ideology often dramatically increases the adoption of products among the software development communities.

Zeller (2007) argues that the future of collaboration is mostly a future of tools, and the more useful, reusable and extensible the tools are the more likely they will be extended into different environments. This was clearly seen in products like Mozilla Prism, or the Google Wave protocol, as their openness quickly allowed developers to create similar tools based on other platforms or discrete requirements. In fact, Zeller puts forward the rationale that 'tomorrow's designers of programming environments should follow the ECLIPSE model and foster integration and collaboration' (Zeller 2007, p.4).

Following this model of integration and collaboration often creates an ecosystem for adoption among the software development community and increases the potential for success. This would facilitate an element of success toward any future collaborative coding environment in the cloud. For this reason, it is recommended that future implementations of the platform look to follow this ideology.

8.3. INTEGRATION AND MASHUPS

Parallel to the concept of extensibility and customization is the growing trend behind integration and mash-up technology popular among recent web applications. The ability to customize the application based on a discrete user's requirements highlights the power of web technologies. Developers often choose to customize or mash-up different tools to tailor a system to meet their needs or architecture.

According to Schroth and Christ (2007) a mash-up is a Web-based resource, be it content or application functionality, which has been created through reuse and composition of two or more different resources.

As at November 2009 ProgrammableWeb.com lists more than 4450 mash-ups, offering customization across a number of different cloud based applications, refer to figure 22 for the most popular across this site. Such technology offers light-weight programming models and protocols which support access to information, increased communication across a number of web platforms and enhanced collaboration among web technologies.

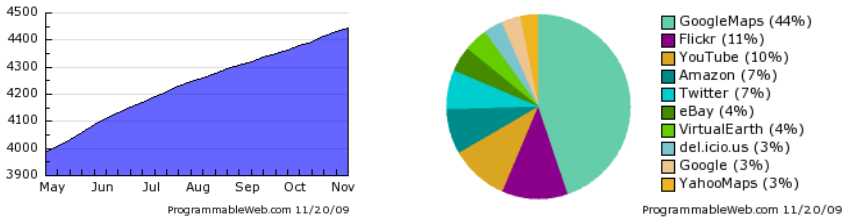


Figure 22 - ProgrammableWeb.com mashup trends

These web technologies provide a comprehensive and global development platform, containing easily usable and integrated resources that can be utilized by individual end-users, as well as across large organizations.

Although not discussed here in detail, the power of mash-ups also facilitates the web's ability to integrate and collaborate across a number of platforms. The rise of these platforms also highlights the emerging trend behind cloud based architecture and

adoption patterns for the future. Future development platforms can look to these concepts in order to achieve similar access to communication and collaboration.

This concept of integration and mash-ups is parallel to the dynamic and scalable model built by Lei et al. (2004), and follows the model proposed by Zeller (2007). By offering access to data, through comprehensive and pluggable components, any web based IDE can easily achieve an extensible plug-in model.

8.4. INFRASTRUCTURE CONCERNS

There are a number of factors to consider based on the proposed infrastructure, including design patterns, performance, costs and availability. The following outlines the rise and fall of AJAX, bandwidth costs and the availability of datacenters. These arguments can be recognized as possible obstructers to the intended platform.

8.4.1. THE RISE OF AJAX

While many of the technologies illustrated throughout this research highlight the benefit behind web 2.0 technologies and AJAX related technologies, there are still a number of pessimistic views in this field. Giglio (2005) states there are a number of issues surrounding accessibility and certain web enabled tools to not cater for sites that rely on large JavaScript functionality.

Gafni (2007) also highlights the concerns AJAX poses around search engine optimization and the inability for search engine robots to index AJAX content. He also argues many web applications have failed at using such technology and designers should not look to AJAX to solve all their interaction and design requirements. Giglio also brings to light concerns on interoperability between browsers and the strong dependencies on JavaScript enabled platforms.

It should be recognized that AJAX is in vogue throughout current web architectures; however such trends may change in the future, thus ultimately affecting the design of the intended platform.

8.4.2. HIGH BANDWIDTH

There are a number of concerns surrounding high bandwidth. Hinchcliffe (2005) brings attention to the argument that AJAX creates the need for fast handling of small messages in a resource intensive manner. Giglio (2005) also notes that JavaScript is not a high performance language, yet web applications are tightly coupled with large and over complicated JavaScript logic. This becomes evident with the need for real-time communication, and is exemplified by the Google Wave architecture, and the asynchronous messages between server, clients, robots, and third party applications.

8.4.3. AVAILABILITY OF DATA CENTERS

Greenberg et al. (2009) coins cloud based data centers as *mega data centers*, due to the increased power and cost requirements maintained through such detailed infrastructure. They states that there is growing concerns surrounding these infrastructures due to the large amounts of fast RAM, large CPU cycles and overbearing disk operations associated in running cloud based data centers (Greenberg et al. 2009).

Greenberg et al. further contends that the more distributed in nature these servers operate, propagation of data between the data centers far-exceed the costs intra-building communication. This becomes relevant given the distributed nature of the intended architecture. They also states that across high delivery data networks costs are in the range of US$18 million per annum, and these infrastructure costs can only be afforded by titanic companies such as Microsoft, Facebook or Yahoo! (Greenberg et al. 2009, pp. 68-69).

Access to and the availability of these resource intensive data centers becomes an area to consider for future implementations. As discussed previously, large organizations such as Google and Amazon have the capacity to host data services in the cloud for both individuals and organizations to host cloud technologies, however the availability of these resources and associated costs plays a crucial role in successful deployment of the intended architecture in the future.

8.4.4. SUMMARY OF INFRASTRUCTURE CONCERNS

While this is not an exhaustive list and concerns are only discussed briefly, they provide areas to consider and any potential concerns that may arise for those researchers looking to deploy the intended framework in the future.

9. FULFILLING A RESEARCH AGENDA FOR DISTRIBUTED COLLABORATION

It should be recognized that the key areas put forward in this research aim at implementing an IDE best suited for cloud based environments of the future. In spite of this, the time invested does not allow for a complete implementation of a collaborative IDE hosted in the cloud. For this reason, a research subset was chosen to address a technical implementation focused on communication and collaboration using existing collaboration technologies. An intelligent agent has been designed and implemented using the Google App Engine and Google Wave architecture. The agent aims to address communication and collaboration in a cloud based environment. The purpose of the agent is to support distributed teams with cross communication and knowledge transfer in an autonomous fashion.

A research agenda was derived from Sengupta, Chandra & Sinha and their 2006 publication titled *A Research Agenda for Distributed Software Development.* According to Sengupta, Chandra & Sinha, 'knowledge sharing between remote team members also gains [research] significance in distributed development, particularly when a new site joins the development effort. Very often, human sources of project-specific information are available, but they may not be known to colleagues at remote sites' (Sengupta, Chandra & Sinha 2006, p. 736).

Sengupta, Chandra & Sinha highlight that there are a number of communication challenges surrounding distributed software development. Moreover, the challenges are particularly concerned with strategic issues pertaining to distribution of work across multiple sites, problems with synchronization and system integration, knowledge management and knowledge sharing, and the re-use of that knowledge across those teams. The team conducted a study at IBM, and found that during distributed development a team from a remote development centre may visit the customer site at the start of the engagement to absorb customer specific knowledge. This knowledge acquisition then needs to be transfer across to the remote centres. However when a team member leaves a project, the important domain and application knowledge the member

has acquired often leaves with him or her. When new team members join the team, they often need to spend significant time acquiring knowledge afresh, before they can begin to start being productive. The IBM participants of this study reported difficulties in gaining a shared understanding of requirements, and in propagating and managing those requirements across the teams. This lead developers to make incorrect assumptions across differing teams, and these discrepancies were hidden until later phases of the project such integration testing.

Sengupta, Chandra & Sinha draw attention to the two major sources of application knowledge. The first as formal knowledge, such as the documentation behind requirements, architecture design, interface specification, code and test cases. The second as informal knowledge, such as ad hoc documents about the application, note gathering onsite, and information persisted to collaboration tools. This notion draws on the fact that informal knowledge is often unique, and untapped, where emails can explain a design rationale, meeting notes, or technical notes surrounding the resolution of an important issue may form some of this informal knowledge.

Through this research synthesis and case study, Sengupta, Chandra & Sinha proposes a research opportunity in order to discover how all such knowledge sources may be integrated to support some kind of virtual assistant that may be embedded within the collaborative tools used by distributed teams. The synthesis goes further, suggesting the example of an assistant that can accept queries, search the knowledge base and find relevant results. If the results are below a certain threshold the assistant can automatically find an appropriate team member to help with the query, route him or her, and keep track if the query has been satisfactorily answered or not. The agent is recognized to provide considerable help for new team members trying to come up to speed.

This research initiative has been the basis for the technical implementation behind an intelligent agent that was constructed. Albeit the agent is a proof of concept, there has been over one hundred hours of design and development spent throughout execution of the agent. The following sections will outline the rationale, implementation, and

suggestions for future implements to the intelligent agent known as Confluxy. Through a number of collaborative platforms the agent can aid developers in cross communication, autonomous up-skill during project delivery, maintain a people finding component and will be crucial for those teams that are globally dispersed where communication channels are not easily achieved, this follows the model proposed by Zeller (2007). The agent gets its name through the definition of conflux, that is, the merge or moving, gathering of forces, people or things. There is also a current trend to end Google Wave robots' names in a 'y'.

9.1. RATIONALE FOR THE AGENT

As mentioned in previous sections, Google Wave provides communication in a real-time collaborative email-like fashion. According to Ben Parr of mashable.com 'It's a hybrid of email, web chat, IM, and project management software. It features the ability to replay conversations because it records the entire sequence of communication, character by character. Because of this, discussions are also live in Google Wave: you will see your friends type character-by-character' (Parr 2009). The Wave architecture allows robots to be developed and participate in Waves they have been invited into, in order to perform a particular purpose. This proved to be an impressive platform to address the requirement for a collaborative tool, and also addressed the requirement for a virtual assistant through its robot architecture.

It is envisioned that organizations who employ large distributed development teams could take advantage of the agent in the following manner:

- A developer has a problem or challenge with the current project and either needs help or up-skill on a topic, or needs assistance from colleagues in order to solve the problem.
- The developer can engage in a Wave with one or more other developers and can converse regarding the problem. When the Wave is initiated the agent is invited to the discussion. In doing so, the developers are inherently asking for support from the agent, and are also willing to potentially help others with the same problems in the future.

- Similar to how email is achieved, the Wave will occupy discussions with any number of parties, concerning any number of topics.
- The agent can assess the discussions occurring, and provide answers for, or sources of information regarding particular topics through interaction and conversations within the Wave.
- The agent then has the potential to manage an index of individuals who discuss particular topics, can suggest to developers other individuals that may be able to assist with the problem, and invite those individuals into the Wave.
- Ultimately, the more Waves the agent is a part of, the more feedback it can receive, the better it learns about key topics to maintain, and those individuals who can provide the highest merit on these topics.

It was recognized that development teams utilize a number of tools such as wikis, issue tracking software, version control systems and other collaboration tools. This provides focus on the sources of knowledge, and the knowledge bases in which the agent would need to acquire information. Consequently, it was determined that the agent could take advantage of the enterprise wiki from Atlassian known as Confluence. The plugin framework embedded within the wiki allowed a custom plugin to be developed to integrate with the internal search framework of the wiki. This initiative, coupled with Confluence's labeling, otherwise known as *tagging*, proved an ideal source of keywords and information available to the agent.

Contrary to the suggests by Sengupta, Chandra & Sinha, there was a crucial decision to have the agent act autonomously in the Wave and not act as an unintelligent and simplistic search engine, as it was perceived that in such cases a developer would simply go to Google to achieve the same task. Instead, the agent will assess messages in the Wave and make decisions based on the content of the message on how to proceed, and whether it should perform search related tasks.

9.2. AGENT ARCHITECTURE

The diagram below shows the basic flow of information and processes throughout the complete architecture.

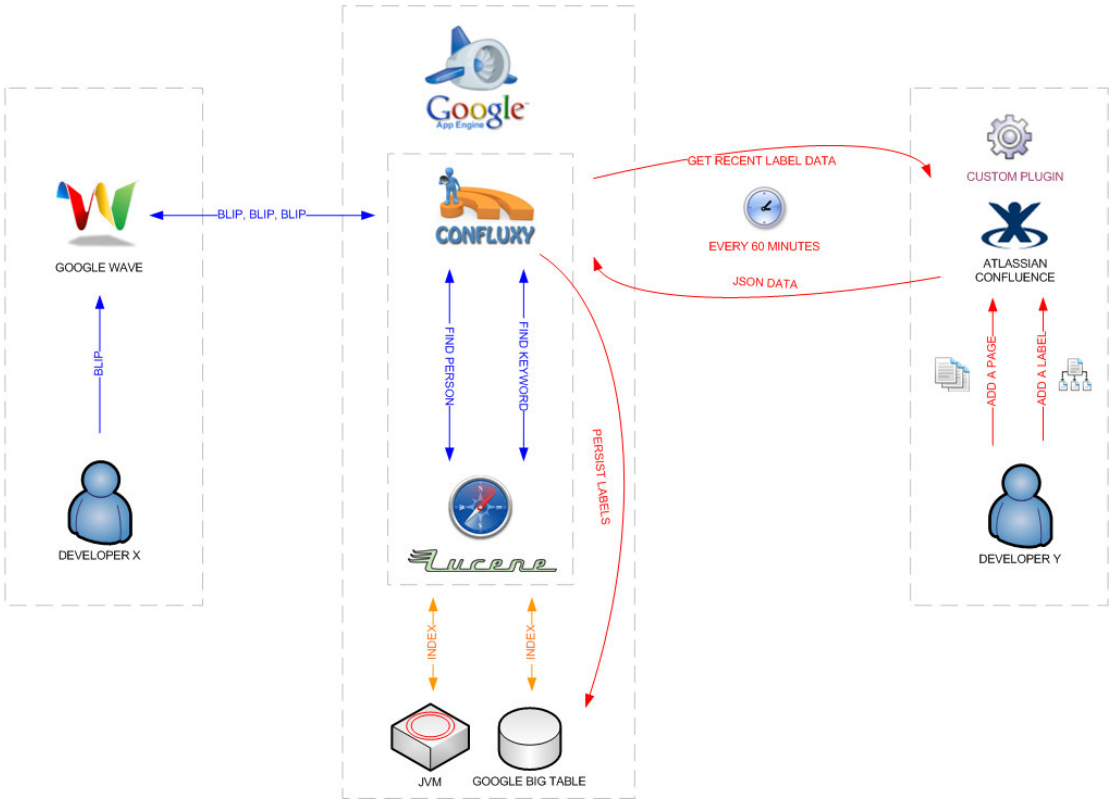


Figure 23 - Agent architecture

The sequence of events that the agent follows is as such:

- Both Developer X and Developer Y are at distributed locations. Both have access to Wave, and both have access to the Confluence wiki.
- Developer Y, along with many other colleagues, add pages and content as necessary, and build up the knowledge base over time.
- Developers can tag content in the wiki with meaningful labels. Tags are arbitrary and can be anything meaningful to the development team. These tags act as the keywords for the agent.
- The Confluxy agent has a cron job which sends a request to the custom defined plugin every 60 minutes.

- During the request the agent authenticates with its own user account which has access to any global information that can be maintained within Wave.
- The custom plugin searches all pages, news items, comments, and content inside attachments, within the workspaces the agent has permission to view. It then formats the data and renders a JSON string.
- The agent receives the response, unmarshals the JSON and persists these results into the Google persistence framework.
- When Developer X requires assistance or information they invite the agent into the Wave. Confluxy then tokenizes each word in each blip, and stores these non-common words in an in-memory index.
- The agent then searches its list of persisted keywords from the wiki labels. If a result is returned it will create a new message which will be appended to the conversation. The message will contain a summary of the content within the wiki associated with the keyword, as well a link to the page in the wiki.
- If the developer provides feedback that agent was helpful, it then persists the developers email and associates them with that keyword for future reference.
- If the user selects they need help from someone else, the agent looks through its data store to find someone associated with that keyword, if someone has been found they ask the developer if they would like that person invited into the Wave.
- If the developer chooses to invite the suggested person, the agent invites this other person into the Wave, and informs the Wave they have done this through an additional message within the Wave.
- Every message the agent creates also has an ignore button attached. If someone selects the ignore button the agent removes the blip from the Wave.

9.3. AGENT IMPLEMENTATION

The agent achieves the intended goals by maintaining two types of indexes. The first is an in-memory index which retains keywords found in a Wave message and stores this information to be accessible in real-time. The second which integrates with the persistence management, and maintains an index of previously persisted keywords sourced from the wiki. The in-memory index is purpose built to index Waves in real-time

and provide relationships between conversations that are being conducted at the same time that are discussing similar topics. This would aim to address any informational overlap across different teams or different individuals. This would address a number of the informational and code conflict challenges addressed previously.

The current working model of the agent removes commons words from Wave messages, commits those keywords to the in-memory index, searches through its persisted keywords for matches, and provides results through pseudo-conversation. When results are displayed, feedback buttons are constructed through the Wave API and presented with the results. If the agent was helpful, it assumes the user is now familiar with this context and can inherently help someone with the same issue in the future. This person will be indexed against the given the keyword for its people search mechanism. If the agent was unhelpful, it looks for users previously associated with this keyword and suggests that they be invited into the Wave. Upon confirmation, the agent will automatically invite the suggested person into the Wave.

Inherent in cloud architecture, the agent is hosted on the Google App Engine (GAE). The agent takes advantage of GAE's persistence management framework, and GAE's cron for its information retrieval process scheduled at predefined intervals. The agent also takes advantage of the Compass framework built on top of Apache Lucene. Through Compass the agent can achieve the analysis of keywords, provide keyword searching and indexing, and support integration into the GAE persistence management framework.

9.4. DEMONSTRATION

Please refer to the Appendix for a detailed set of instructions on how to use the agent.

9.5. SOURCE CODE

For the Google Wave code refer to – http://thesis.domlovell.com/agent/confluxy.zip
For the Confluence connector code refer to – http://thesis.domlovell.com/agent/confluxy-confluence.zip

9.6. FUTURE DIRECTION

The agent currently maintains intelligence through its use of the underlying frameworks to parse language and generate matches. There is a high dependency on the Compass framework in order to achieve the required goals. The framework is also not used to its full capacity and only performs basic analysis of sentences. There also lacks appropriate reasoning and ability to rate and score keywords, which would determine whether certain keywords should be searched and the weighting behind those keywords. An accurate contextual agent would also need to support semantics and natural language parsing, while maintaining a complex indexing and ranking system. The agent would need to intelligently be able to increase the ranking of particular topics the more they are discussed. Future implementations of the same virtual assistance would need to look at how the analysis of natural language can be achieved, as well as implementing a scoring system for keywords or contextual information.

There is also an intersection between true autonomy and the event driven processes the agent inhibits. The agent can only learn about user-keyword associations when users engage in pressing buttons, it can then learn which people react to different results. Improved autonomy is an area of study future researchers must aim to address.

If Google Wave is the intended platform for future implementations it can be foreseen that data sources could be dynamically loaded during a Wave. For example, a sub-Wave could be initiated to provide a widget or configuration screen, where the source of data could be implemented on demand.

It is recommended that future implementations of the agent source data from multiple locations, or any arbitrary location that would be relevant to the development team. In such multiple data source architectures, smaller purpose built agents can be deployed through each end point and each agent can negotiate on the information that is to be retained or extracted for the major agent. Integration with an agent that handled LDAP would add increase support for the process of finding information about other teams or individuals, and provide extra information such as extension numbers or roles within the company.

Figure 24 - Future agent implementations

Possible future end points could be Google, Wikipedia, Atlassian JIRA, Github, Twitter, Glue or True Knowledge. Using the Google REST API real-time searches could be conducted on the Google search engine. Wikipedia also offer a public search API, this could provide information to a large knowledge pool.

Development team's utilizing the Atlassian JIRA issue tracker could take advantage of the plugin framework and search existing bugs and solutions with the development team's issue tracker. Github, the collaborative and social version control system, offers an API for searching projects with keywords or particular names. This would be of benefit to those teams already using Github to host source code. The real-time Twitter search API would allow you to find other developers, both internal to an organization using Twitter lists, and externally using the entire feed, and find people who are discussing particular keywords or topics. This would be easily achieved through Twitters hash tag annotations.

The intelligence system known as Glue could be utilized for semantic people recognition, and finding individuals with or who are interested in a particular topic, which would aid the people finding component of this agent. The True Knowledge semantic search engine and its public API would also greatly assist the natural language parsing that would need to be coupled with future implementations of this agent.

9.7. SUMMARY OF THE AGENT

The agent provides a significance addition to the research of Sengupta, Chandra, and Sinha (2006), while also allowing a considerable breadth of direction in this field of research for those wishing to address this research agenda. It addresses the question on how developers can take advantage of cloud based architectures in a distributed nature to support communication and collaboration, using technologies currently available.

10. RESEARCH OUTCOME

The purpose of this research posed the question of whether an IDE can be deployed using cloud based architecture, and whether the software development community would successfully adopt such an environment.

While a number of existing cloud based platforms are established such as the Amazon S3 platform and the Google App engine as described in previous sections, there are a number of factors to consider whether a collaborative development environment is possible based on technologies currently available. This research has outlined the argument that cloud computing has not yet reached it's full potential given its infancy, as well as a number of infrastructure concerns, which may inhibit the realization of this platform. The qualitative survey provides an insight into the opinions of industries professionals. While one cannot generalize from this survey alone, based on the data that was collected there was a clear division between the parties for and parties against a cloud based IDE. The extended response questions provided a rationale for many of these concerns, most of which where concerns with security or the threat of data loss.

A look into the direction industry leaders were headed was outlined with the case studies behind Google Wave and Mozilla Bespin. Although these products are yet to reach their full potential, it could be argued that there is a significant amount of work that is still required from the products. Given the scale from which these organizations operate also provides an indication into the technical resources and financial requirements necessary for this research.

As put forward by Woodward and Hallett (2008) the industry has only witnessed the beginning of cloud computing. Therefore based on the aforementioned factors, this research concludes that due to available technologies, demand for the environment, and infrastructure concerns, a collaborative IDE hosted in the cloud cannot be released in the immediate future. Those researchers wishing to do so must reassess the industry and the potential to realize such a project in the following years once the platform has matured.

11. CONCLUSION AND FUTURE RESEARCH

This research was focused on development environments of the future and posed the question of whether a collaborative IDE can be hosted using cloud based environments. A number of challenges with existing development practices were raised, including the push for global teams and their need for transparency, a lack of synergy between project components, and challenges for developers on the move. Based on these challenges a hypothesis was made on cloud computing and the indented framework necessary in overcoming these challenges. This aimed at contributing to a number of gaps in current research arenas concerning collaboration in the software engineering community.

Significant research was conducted on individuals in the research community who have aimed at addressing a number of these challenges, such as the Jazz framework, the toolet system for JEdit, Moomba the collaboration platform, a collaborative web browser and an IDE for a mobile device. These research agenda gave light on the need for future research.

A number of pioneering platforms, including Google Wave and Mozilla Bespin were looked at in detail, which provides a case study on how industry leaders are attempting to solve these challenges. It was deemed there were a number of factors concerned with these products that were yet to be realized, although the roadmap shows potential.

Through a qualitative study this research was able to determine the needs of a subset of industry professionals and their opinions of the research proposition. A customized survey allowed aggregate data to be collected on those professionals and well as extended responses on their views on a number of topics. Key responses provided a rationale behind the discontent respondents had toward cloud computing.

An intended framework was proposed, which aimed at mirroring concepts traditionally found in social networks, to offer developers social transparency and embedded features

to support seamless communication between teams. A target audience was outlined, as well as a number of suggestions for success.

A number of facilitators and obstructers were discussed, and how they will impact the nature of the cloud computing industry in the coming years. These included emerging technologies, adoption patterns, and infrastructure concerns.

Given the limited time and technical resources, it was deemed that the research period would not allow a complete deployment of the intended platform. As such, a research subset was chosen and implemented. This included an intelligent agent built on top of the Google cloud infrastructure and the Google Wave collaboration medium. A number of potential improvement and further research conditions were also raised based on this research agenda.

Conclusively the research outcome expressed why there were a number of factors that prevent the immediate realization of the intended platform. However, this would only be limited by the growth of cloud computing and perceptions surrounding the technology in the future.

It is recommended that future research should aim to focus on the adoption patterns behind cloud computing, and the underlying success or decline of the platform. Those individuals aiming to provide extensive research behind communication methods must also address any cultural problems surroundings the communication mediums, which has not be significantly addressed through this research. It is also recommended that those researchers looking to implement the proposed framework should follow the factors for success and ensure the platform has seamless integration between other development tools.

This research concludes that the chosen model is not yet mature enough to be realized. Based on this there are three major streams of research that individuals can extend from the research. These include the implementation of the proposed model upon industry

maturity, the suggested improvements to the communication and collaboration agent developed, or a complete reassessment of the philosophy that has been proposed through the research.

Regardless of the future outcome, this research not only provides a significant step towards improving communication and collaboration throughout the software development communities, but provides an umbrella concept around these models for other industries to follow. These may include education industries and universities looking to deploy an improved software teaching model, or efforts to support improved group collaboration among students.

Development transparency through the power of SaaS and social networking offers a paradigm shift to the way software development is achieved. Implementation of such a tool can allow up to the minute information about people and the work being completed around you. Future researchers must look toward addressing barriers behind cloud computing, including security and data reliability. In do so, researchers can implement a safe and extensible framework that is backed by cloud based architecture.

12. REFERENCES

Almaer, D. 2009, *Bespin: A new Mozilla Labs experimental extensible code editor using Canvas*, Ajaxian.com, viewed March 31st 2009, <http://ajaxian.com/archives/bespin-a-new-mozilla-labs-experimental-extensible-code-editor-using-canvas>

Atlassian 2009, *Continuous Integration Build Tool – Bamboo*, Atlassian Sydney Australia, viewed 1st September 2009, <http://www.atlassian.com/software/bamboo/>

Beck, K., Andres, C. 2004, *Extreme Programming Explained: Embrace Change*, Addison-Wesley Professional, 2nd edn.

Bellotti, V. & Bly, S. 1996, 'Walking away from the desktop computer: distributed collaboration and mobility in a product design team', paper presented to the *Proceedings of the 1996 ACM conference on Computer supported cooperative work*, Boston, Massachusetts, United States.

Broersma, M. 2009, *Mozilla shifts code development to the cloud*, zdnetasia.com, viewed March 30th 2009, <http://www.zdnetasia.com/news/software/0,39044164,62051132,00.htm?scid=rss_z_nw>

Bruce, G., Robson, P. & Spaven, R. 2006, 'OSS opportunities in open source software -- CRM and OSS standards', BT Technology Journal, vol. 24, no. 1, pp. 127-140.

Burke M. 2004, 'Proceedings of the 2004 OOPSLA workshop on eclipse technology eXchange', ACM, Vancouver, British Columbia, Canada, p. 105.

Cheng, L.-T., Hupfer, S., Ross, S. & Patterson, J. 2003, 'Jazzing up Eclipse with collaborative tools', paper presented to the *Proceedings of the 2003 OOPSLA workshop on eclipse technology eXchange*, Anaheim, California, pp. 45-49.

Clayberg, E. & Rubel, D. 2006, Eclipse: Building Commercial-Quality Plug-ins (2nd Edition) (Eclipse), Addison-Wesley Professional.

Cohen, T. & Clemens, B. 2005, 'Social networks for creative collaboration', paper presented to the *Proceedings of the 5th conference on Creativity & cognition*, London, United Kingdom, pp. 252-255.

Connaghan, D.P 2008, 'Accessibility in Rich Internet Applications', *Dublin Institute of Technology, School of Computing,* Thesis, p. 31.

Damian, D., Lanubile, F. & Mallardo, T. 2006, 'The role of asynchronous discussions in increasing the effectiveness of remote synchronous requirements negotiations', *paper presented to the Proceedings of the 28th international conference on Software engineering*, Shanghai, China, pp. 917-920.

de Souza, C.R.B., Redmiles, D. & Dourish, P. 2003, '"Breaking the code", moving between private and public work in collaborative software development', paper presented to the *Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work*, Sanibel Island, Florida, USA, pp. 105-114.

de Souza, C.R.B., Redmiles, D., Cheng, L.-T., Millen, D. & Patterson, J. 2004, 'How a good software practice thwarts collaboration: the multiple roles of APIs in software development', paper presented to the *Proceedings of the 12th ACM SIGSOFT twelfth international symposium on Foundations of software engineering*, Newport Beach, CA, USA, pp. 221-384.

Ditchendorf, T. 2007, *Fluid - Free Site Specific Browser for Mac OS X Leopard*, FluidApp.com, viewed 30th August, <http://fluidapp.com>

Eclipse Foundation 2009, *Mylyn,* The Eclipse Foundation, viewed 1st September 2009, < http://www.eclipse.org/projects/project_summary.php?projectid=tools.mylyn>

Ferraté A. 2009, *An Introduction to Google Wave - Google Wave: Up and Running*, O'Reilly Publishing, Oregon, viewed 10th October 2009 <http://www.oreillynet.com/pub/a/webdevelopment/excerpts/9780596806002/google-wave-intro.html>

Gafni, E. 2007, 'Coming clean with AJAX', *interactions*, vol. 14, no. 6, pp. 22-22.

Galbraith, B., Almaer, D. 2009, *Introducing Bespin*, Mozilla, viewed 12th February 2009, <https://mozillalabs.com/blog/2009/02/introducing-bespin/>

Gamma E., Beck, K. 2003, *Contributing to Eclipse: Principles, Patterns, and Plugins*, Addison Wesley Longman Publishing Co., Inc., Redwood City, CA.

Giglio, J. 2005, 'AJAX: Highly Interactive Web Applications', *Running head: AJAX*, pp. 1-9.

Google 2009, *Google Wave Preview,* Google.com, viewed 1st June 2009, <http:// wave.google.com/>

GoogleCode 2009, *Sametimed*, Google, viewed 12th Oct 2009, <http://code.google.com/p/sametimed/>

Greenberg, S., Gutwin, C., and Cockburn, A. 1996, 'Using distortion-oriented displays to support workspace awareness', paper for the *Proceedings of the HCI'96 People and Computers XI*, Dept of Comp. Science, Univ. of Calgary, Canada.

Greenberg, A., Hamilton, J., Maltz D.A., Patel, P. 2009, 'The Cost of a Cloud: Research Problems in Data Center Networks', *ACM SIGCOMM Computer Communication Review,* Microsoft Research, Redmond, WA, USA, vol. 39, no. 1, pp. 68-73.

Grossman, F., Bergin, J., Leip, D., Merritt, S. & Gotel, O. 2004, 'One XP experience: introducing agile (XP) software development into a culture that is willing but not ready', *paper presented to the Proceedings of the 2004 conference of the Centre for Advanced Studies on Collaborative research*, Markham, Ontario, Canada.

Hinchcliffe, D. 2005, *State of Ajax: Progress, Challenges, and Implications for SOAs*, hinchcliffe.org, viewed 30th October 2009 ,<http://hinchcliffe.org/archive/2005/08/ 18/1675.aspx>

Hinchcliffe, D. 2009, *The enterprise implications of Google Wave*, Zdnet.com, viewed 1st November 2009, <http://blogs.zdnet.com/Hinchcliffe/?p=400>

Herbsleb, J.D. 2007, 'Global Software Engineering: The Future of Socio-technical Coordination', paper presented to the *2007 Future of Software Engineering*, pp. 188-198.

Herrman, J. 2009, 'Everything You Need To Know About Chrome OS', Gizmodo.com, viewed 19th November 2009, <http://gizmodo.com/5408504/everything-you-need-to-know-about-chrome-os>

Hopkins, M. 2008, *Can We Please Define Cloud Computing?*, Mashable.com, viewed 2nd November 2009, <http://mashable.com/2008/08/19/cloud-computing-defined/>

Jarvensivu, J., Kosola, M., Kuusipalo, M., Reijula, P. & Mikkonen, T. 2006, 'Developing an Open Source Integrated Development Environment for a Mobile Device', *Software Engineering Advances, International Conference on*, pp. 55-57.

Joseph-Malherbe, S. and Malherbe, D. 2009, 'Manage changes in the requirements definition through a collaborative effort', paper presented to the *6th Annual INCOSE SA Conference, CSIR ICC,* Pretoria, South Africa, pp. 1-10.

Koskinen, T. 2006, 'Social software for industrial interaction', paper presented to the *Proceedings of the 18th Australia conference on Computer-Human Interaction: Design: Activities, Artefacts and Environments*, Sydney, Australia, pp. 381-384.

Lei, H., Chakraborty, D., Chang, H., Dikun, M.J., Heath, T., Li, J.S., Nayak, N., Patnaik, Y. 2004, 'Contextual Collaboration: Platform and Applications' , paper presented to *the 2004 IEEE International Conference on (SCC'04),* pp.197-206.

Miller M. 2008, *Cloud Computing – Web-Based Applications That Change the Way You Work and Collaborate Online*, Que Publishing, Pearson Education, Canada.

Mozilla 2009, *Bespin >> Code In The Cloud,* Mozilla, viewed 1st March 2009, <https://bespin.mozilla.com/>

Mozilla Labs 2007, *Introducing Prism*, Mozilla, viewed 30th August 2009, <http://mozillalabs.com/blog/2007/10/prism/>

Olson, G.M. and Olson, J.S. 2000, 'Distance Matters.', *Human-Computer Interaction*, vol. 15, pp. 139-178.

Paul, R. 2009, *Mozilla's Bespin project encourages experimentation*, Arstechnica.com, viewed 31st March 2009, <http://arstechnica.com/open-source/news/2009/02/mozillas-bespin-project-encourages-experimentation.ars>

Parr, B. 2009, *Could Google Wave Redefine Email and Web Communication?*, Mashable.com, viewed 4th September 2009, <http://mashable.com/2009/05/28/google-wave/>

Reeves, M. & Zhu, J. 2004, 'Moomba – A Collaborative Environment for Supporting Distributed Extreme Programming in Global Software Development', *Extreme Programming and Agile Processes in Software Engineering*, pp. 38-50.

Reimer, J. 2007, *Google "gears" up for offline web apps with new API, offline Reader*, ArsTechnica.com, viewed August 25th 2009, <http://arstechnica.com/old/content/2007/05/google-gears-up-for-offline-web-apps-with-new-api.ars>

Sarma, A., Noroozi, Z., and Hoek, A.V.D. 2003, 'Palantír: raising awareness among configuration management workspaces'. in *International Conference on Software Engineering*. Portland, Oregon, pp. 444-453.

Schroth, C. and Christ, O. 2007, 'Brave New Web: Emerging Design Principles and Technologies as Enablers of a Global SOA', in *Proceedings of the IEEE International Conference on Services Computing*, St.Gallen, Switzerland, p. 2.

Schwartz, B. 2007, *Google Gears Brings Offline Web Applications To Life,* Searchengineland.com, viewed August 25th 2009, <http://searchengineland.com/google-gears-brings-offline-web-applications-to-life-11342>

Sengupta, B., Chandra, S. & Sinha, V. 2006, 'A research agenda for distributed software development', *paper presented to the Proceedings of the 28th international conference on Software engineering*, Shanghai, China.

Sengupta, C., Papakipos, M. 2009, *Releasing the Chromium OS open source project*, Google, viewed 19th November 2009, <http://googleblog.blogspot.com/2009/11/releasing-chromium-os-open-source.html>

Siebeck, R., Janner, T., Schroth, C., Hoyer, V., Wörndl, W., & Urmetzer, F. 2009, 'Cloud-based Enterprise Mashup Integration Services for B2B Scenarios', publication for the *2nd Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009) in conjunction with the 18th International World Wide Web Conference (WWW2009).*

Stewart, K.J., Darcy, D.P. & Daniel, S.L. 2005, 'Observations on patterns of development in open source software projects', paper presented to the *Proceedings of the fifth workshop on Open source software engineering*, St. Louis, Missouri.

Trampoline Systems 2009, *Solutions powered by SONAR*, Trampoline Systems, London, UK, viewed 1st September 2009, <http://www.trampolinesystems.com /solutions>

Vander Wal, T. 2007, *Folksonomy Coinage and Definition*, vanderwal.net, viewed 30th August 2009, <http://vanderwal.net/folksonomy.html>

Whitehead, J. 2007, 'Collaboration in Software Engineering: A Roadmap', paper presented to the *2007 Future of Software Engineering*, pp. 214-225.

Woodward, J., Hallett, D. 2008, 'THE FUTURE OF ENTERPRISE COMPUTING IN 2015', Architecting The Future, weblog, viewed 11 June 2009, <http://darrenhallett.wordpress.com/2008/06/06/enterprise-computing-in-2015/>

Yap, N., Chiong, H.C., Grundy, J. & Berrigan, R. 2005, 'Supporting Dynamic Software Tool Integration via Web Service-Based Components', paper presented to the *Proceedings of the 2005 Australian conference on Software Engineering*, pp. 160-169.

Zeller, A. 2007, 'The Future of Programming Environments: Integration, Synergy, and Assistance', paper presented to the *2007 Future of Software Engineering*, pp. 316-325.

13. APPENDIX

13.1. SURVEY QUESTIONS

Section One

1 Industry, Role, Education and Experience

1.) Country:
Please Select

2.) Industry Sector:
Please Select

3.) Job Title:
Please Select

5.) Educational Level:
Please select

6.) Years of experience in current role:
Please select

Section Two

2 Development Style and General Practices

7.) The development methodology I follow is best described as:
- ○ Agile
- ○ Scrum
- ○ Extreme Programming
- ○ Rational
- ○ CMMI
- ○ Waterfall
- ○ Test Driven
- ○ None of the above

8.) I use the following communication tools during development:
- ☐ IM
- ☐ Blogs
- ☐ Forums
- ☐ Shared Calendars
- ☐ Workflow Systems
- ☐ Document Management Systems
- ☐ Wikis
- ☐ Issue tracking software

9.) My development team consists of:
- ○ Just myself
- ○ 2 or more people
- ○ 5 or more people
- ○ 10 or more people
- ○ 15 or more people

9a.) Does your team operate under a distributed environment?:

☐ Yes

10.) The majority of my software development is completed using the following Integrated Development Environment (IDE):

- ○ Eclipse (any version)
- ○ Visual Studio (any version)
- ○ IntelliJ
- ○ Netbeans
- ○ Delphi
- ○ KDevelop
- ○ Komodo
- ○ Aptana Studio
- ○ Bespin
- ◉ None of the above

Please specify:

10a.) Why do you use this IDE?

11.) The organization I am apart of can be best described as the following management style:

If you are self employed choose the most appropriate management style you follow

- ○ Autocratic
- ○ Paternalistic
- ○ Democratic
- ○ Laissez-faire
- ○ Empowered Workers
- ◉ None of the above

Please specify:

Section Three

3 Developer Satisfaction

12.) My current IDE performs all the tasks I expect of it:

○ Strongly Disagree ○ Disagree ◉ Neutral ○ Agree ○ Strongly Agree

12.a) What would you change with your existing IDE?

13.) I regularly seek help from others during development

○ Strongly Disagree ○ Disagree ◉ Neutral ○ Agree ○ Strongly Agree

13a.) Please describe any tools you find facilitate communication among others during development

14.) Development teams must operate in close proximity to one another in order to be successful

○ Strongly Disagree ○ Disagree ◉ Neutral ○ Agree ○ Strongly Agree

15.) Pair-programming aids development processes

○ Strongly Disagree ○ Disagree ◉ Neutral ○ Agree ○ Strongly Agree

16.) My team manages communication effectively

○ Strongly Disagree ○ Disagree ◉ Neutral ○ Agree ○ Strongly Agree

16a.) What is the most effective tool you use for communication?

17.) Knowledge and expertise is shared among my team

○ Strongly Disagree ○ Disagree ◉ Neutral ○ Agree ○ Strongly Agree

18.) I am constantly aware of what others are doing

○ Strongly Disagree ○ Disagree ◉ Neutral ○ Agree ○ Strongly Agree

19.) I understand my colleagues design decisions

○ Strongly Disagree ○ Disagree ◉ Neutral ○ Agree ○ Strongly Agree

20.) Development teams benefit from the use of social and collaborative tools during project delivery

○ Strongly Disagree ○ Disagree ◉ Neutral ○ Agree ○ Strongly Agree

20a.) Do you use social networking tools to facilitate development? If yes, please describe how and why you use them.

21.) My team promotes independence with design and development phases

○ Strongly Disagree ○ Disagree ◉ Neutral ○ Agree ○ Strongly Agree

22.) Code conflicts occur regularly in my team

○ Strongly Disagree ○ Disagree ◉ Neutral ○ Agree ○ Strongly Agree

22a.) Describe the impact this has on late or unsuccessful project

23.) I would like an IDE available that requires less configuration and setup then my existing IDE

○ Strongly Disagree ○ Disagree ◉ Neutral ○ Agree ○ Strongly Agree

24.) I would prefer to have my IDE hosted in the cloud, rather then on my workstation

○ Strongly Disagree ○ Disagree ◉ Neutral ○ Agree ○ Strongly Agree

24a.) What are your predictions on the likelihood of mainstream cloud-based IDEs in the near future?

24b.) Discuss any barriers you predict may prevent this from happening

Section Four

4 Survey Confirmation

25.) Would you like to be sent a copy of the results?

Yes: ☑

Email Address:

An email will be sent to you shortly informing you of the report you will receive, once all results have been collected.

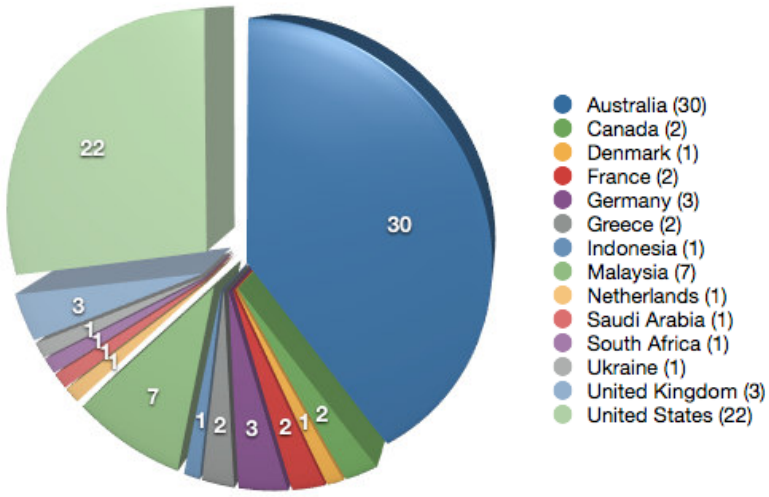
26.) Are you **human**? Enter the two words below:

TABULAR AND GRAPHICAL RESPONSES

13.1.1. SECTION ONE - INDUSTRY, ROLE, EDUCATION AND EXPERIENCE

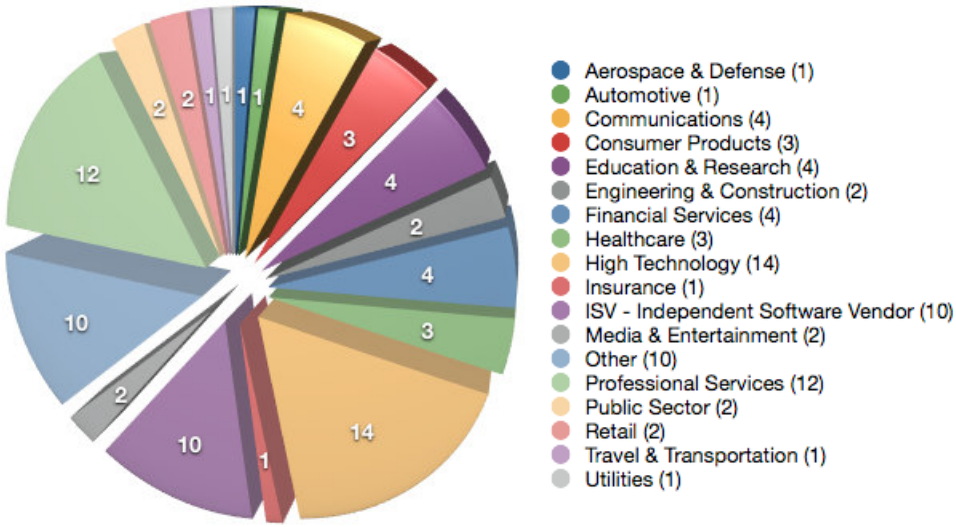
1.) Country:

Country	Tally
Australia	30
Canada	2
Denmark	1
France	2
Germany	3
Greece	2
Indonesia	1
Malaysia	7
Netherlands	1
Saudi Arabia	1
South Africa	1
Ukraine	1
United Kingdom	3
United States	22



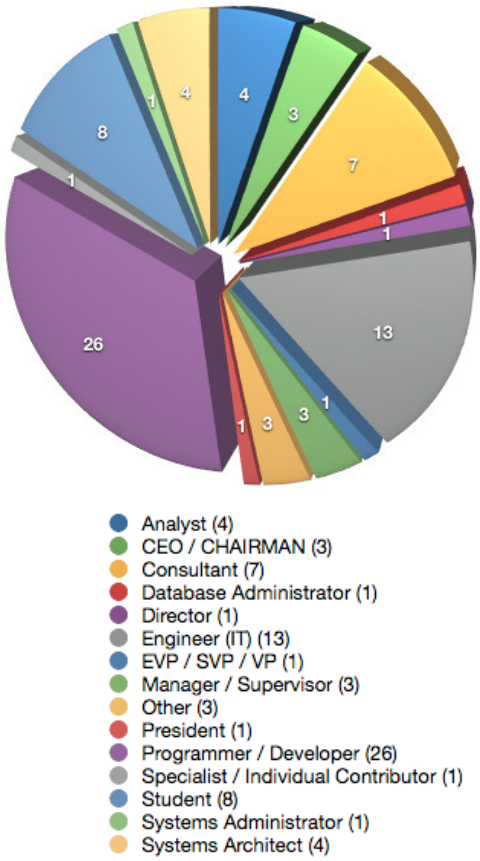
2.) Industry Sector:

Industry	Tally
Aerospace & Defense	1
Automotive	1
Communications	4
Consumer Products	3
Education & Research	4
Engineering & Construction	2
Financial Services	4
Healthcare	3
High Technology	14
Insurance	1
ISV - Independent Software Vendor	10
Media & Entertainment	2
Other	10
Professional Services	12
Public Sector	2
Retail	2
Travel & Transportation	1
Utilities	1



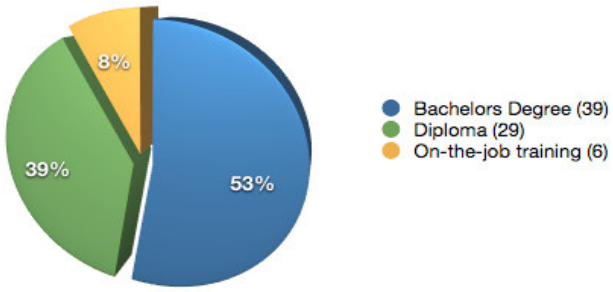
3.) Job Title:

Role	Tally
Analyst	4
CEO / CHAIRMAN	3
Consultant	7
Database Administrator	1
Director	1
Engineer (IT)	13
EVP / SVP / VP	1
Manager / Supervisor	3
Other	3
President	1
Programmer / Developer	26
Specialist / Individual Contributor	1
Student	8
Systems Administrator	1
Systems Architect	4



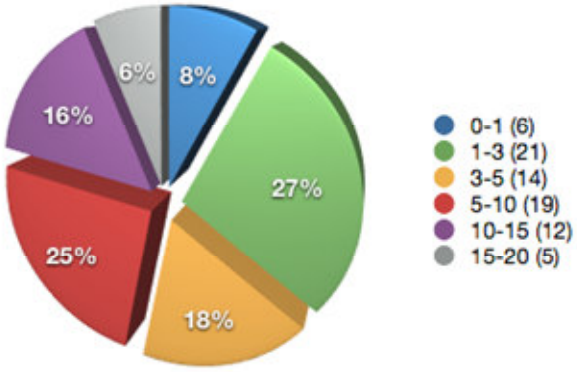
5.) Educational Level:

Education	Tally
Bachelors Degree	39
Diploma	29
On-the-job training	6



6.) Years of experience in current role:

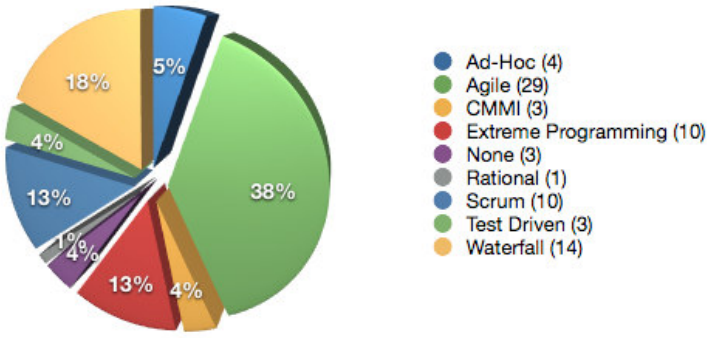
Years	Tally
0-1	6
1-3	21
3-5	14
5-10	19
10-15	12
15-20	5



13.1.2. SECTION TWO - DEVELOPMENT STYLE AND GENERAL PRACTICES

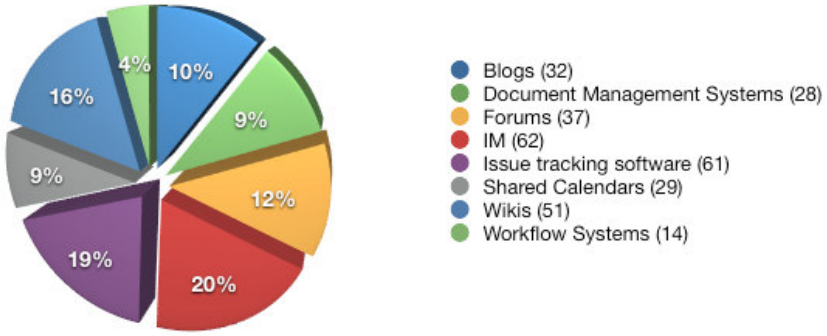
7.) The development methodology I follow is best described as:

Methodology	Tally
Ad-Hoc	4
Agile	29
CMMI	3
Extreme Programming	10
None	3
Rational	1
Scrum	10
Test Driven	3
Waterfall	14



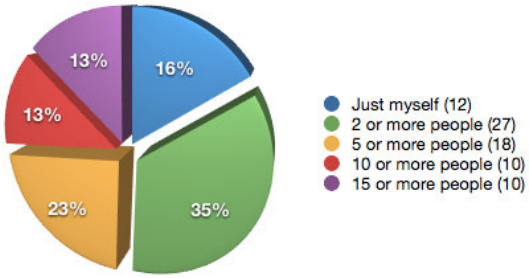
8.) I use the following communication tools during development:

Tool	Tally
Blogs	32
Document Management Systems	28
Forums	37
IM	62
Issue tracking software	61
Shared Calendars	29
Wikis	51
Workflow Systems	14



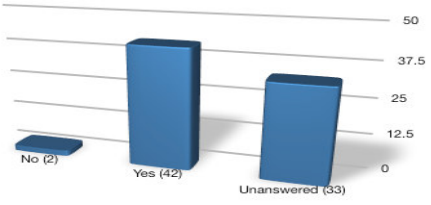
9.) My development team consists of:

Size	Tally
Just myself	12
2 or more people	27
5 or more people	18
10 or more people	10
	10



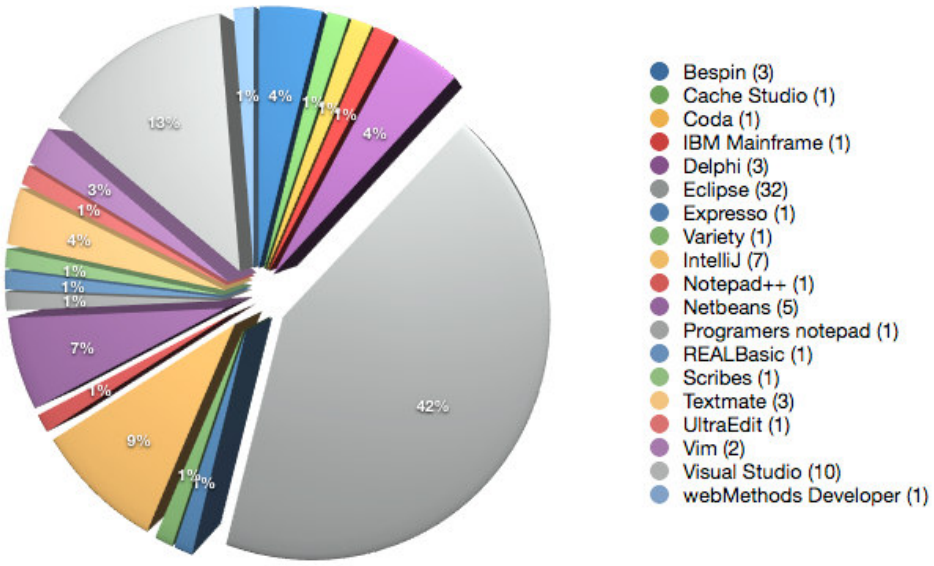
9a.) Does your team operate under a distributed environment?:

Answer	Tally
No	2
Yes	42
Unanswered	33



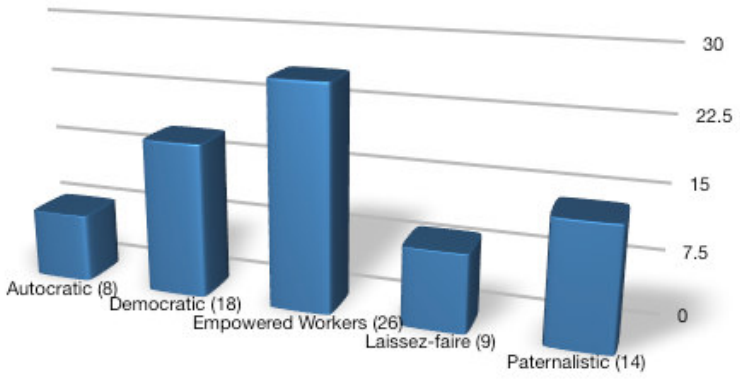
10.) The majority of my software development is completed using the following Integrated Development Environment (IDE):

IDE	Tally
Bespin	3
Cache Studio	1
Coda	1
IBM Mainframe	1
Delphi	3
Eclipse	32
Expresso	1
Variety	1
IntelliJ	7
Notepad++	1
Netbeans	5
Programers notepad	1
REALBasic	1
Scribes	1
Textmate	3
UltraEdit	1
Vim	2
Visual Studio	10
webMethods Developer	1



11.) The organization I am apart of can be best described as the following management style:

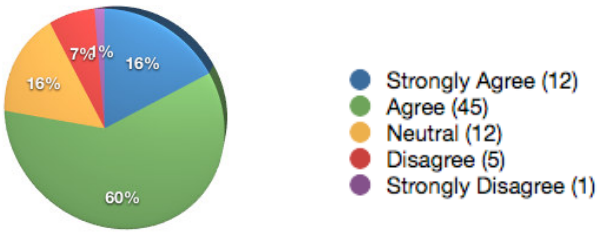
Management Style	Tally
Autocratic	8
Democratic	18
Empowered Workers	26
Laissez-faire	9
Paternalistic	14



13.1.3. SECTION THREE - DEVELOPER SATISFACTION

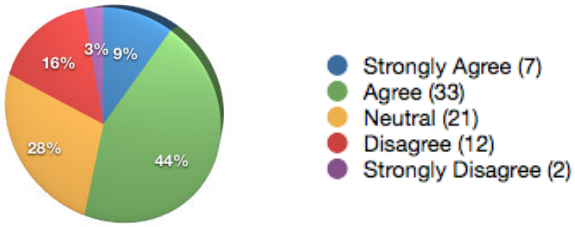
12.) My current IDE performs all the tasks I expect of it:

Strongly Agree	12
Agree	45
Neutral	12
Disagree	5
Strongly Disagree	1



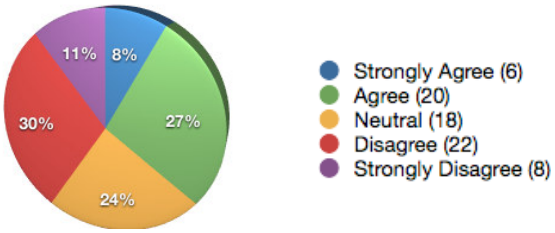
13.) I regularly seek help from others during development:

Strongly Agree	7
Agree	33
Neutral	21
Disagree	12
Strongly Disagree	2



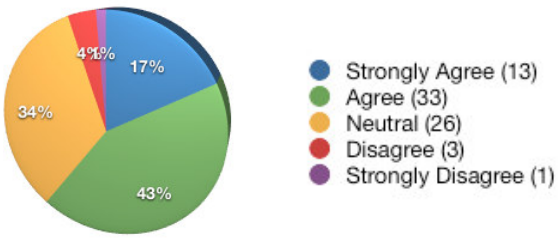
14.) Development teams must operate in close proximity to one another in order to be successful

Strongly Agree	6
Agree	20
Neutral	18
Disagree	22
Strongly Disagree	8



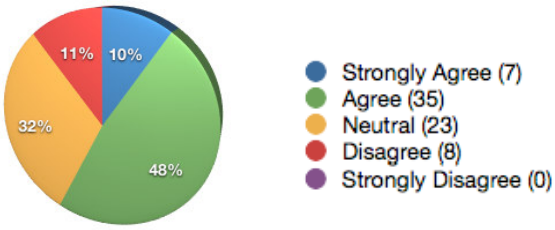
15.) Pair-programming aids development processes

Strongly Agree	13
Agree	33
Neutral	26
Disagree	3
Strongly Disagree	1



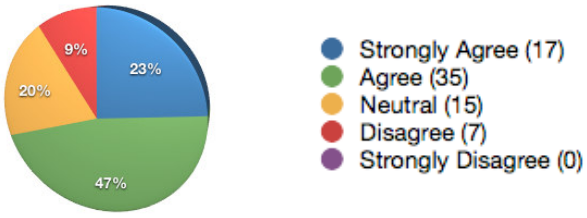
16.) My team manages communication effectively

Strongly Agree	7
Agree	35
Neutral	23
Disagree	8
Strongly Disagree	0



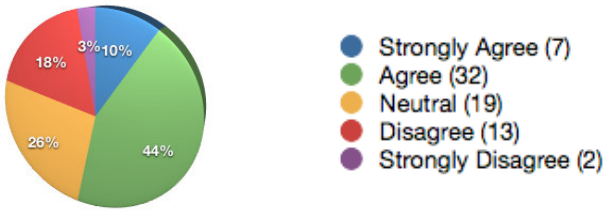
17.) Knowledge and expertise is shared among my team

Strongly Agree	17
Agree	35
Neutral	15
Disagree	7
Strongly Disagree	0



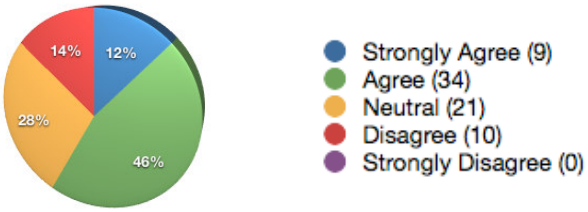
18.) I am constantly aware of what others are doing

Strongly Agree	7
Agree	32
Neutral	19
Disagree	13
Strongly Disagree	2



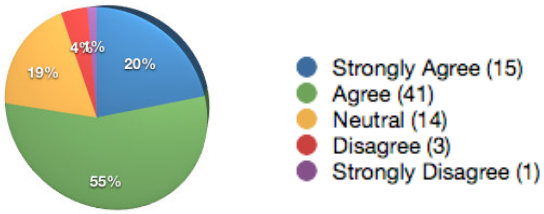
19.) I understand my colleagues design decisions

Strongly Agree	9
Agree	34
Neutral	21
Disagree	10
Strongly Disagree	0



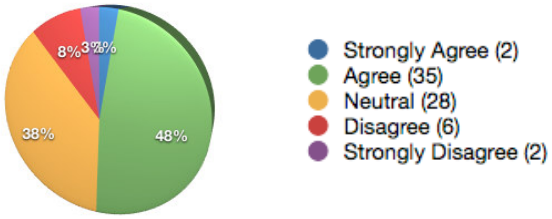
20.) Development teams benefit from the use of social and collaborative tools during project delivery

Strongly Agree	15
Agree	41
Neutral	14
Disagree	3
Strongly Disagree	1



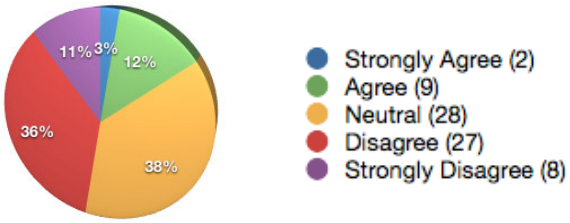
21.) My team promotes independence with design and development phases

Strongly Agree	2
Agree	35
Neutral	28
Disagree	6
Strongly Disagree	2



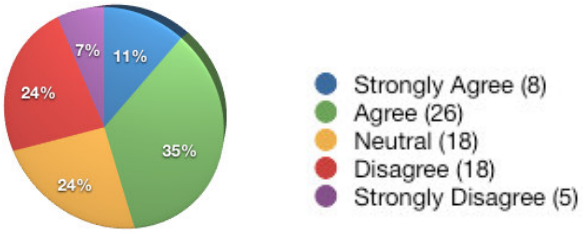
22.) Code conflicts occur regularly in my team

Strongly Agree	2
Agree	9
Neutral	28
Disagree	27
Strongly Disagree	8



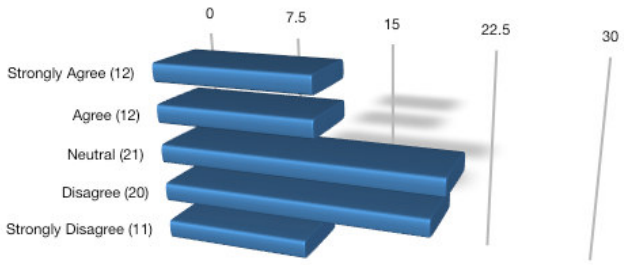
23.) I would like an IDE available that requires less configuration and setup then my existing IDE

Strongly Agree	8
Agree	26
Neutral	18
Disagree	18
Strongly Disagree	5



24.) I would prefer to have my IDE hosted in the cloud, rather then on my workstation

Strongly Agree	12
Agree	12
Neutral	21
Disagree	20
Strongly Disagree	11



13.1.4. EXTENDED RESPONSE QUESTIONS

10a.) Why do you use this IDE?
in response to 10.) The majority of my software development is completed using the following Integrated Development Environment (IDE):

- Textmate
  - Doesn't get in my way

- IntelliJ
  - It's the best one out there for Java development that money can buy. Excellent refactoring support and out-of-the-box functionality is complete.

- Vim + maven (when compiling java)
  - I hate IDEs. This is a very nice text editor. Its keybindings and abilities make it very easy to edit text quite quickly.

- Eclipse
  - Because I am developing (a part of) Eclipse.

- Eclipse
  - Because it provides the best support for AS3, the language we do most of our development in.

- UltraEdit
  - Because it's fast and has logical key shortcuts

- Netbeans
  - Simple, works well for PHP, Java and HTML/CSS/JS
  - Automatic code formatting (for whatever reason this is huge for me)

- Netbeans
  - Easy to figure out how to do things.

- Eclipse
  - This IDE is simple to use and does not requires a lot of memory usage compared with IntelliJ and Netbeans,

- Eclipse
  - It generally does what I need.
  - It's free.

- Delphi
  - We build Rich-client apps for Windows. There is still nothing better for this type of app.

- IntelliJ
  - Company standard

- Eclipse
  - My team uses it - and it's not bad. I don't care, really, as long as there's syntax highlighting and code autocomplete! But then I'm not a serious developer.

- Eclipse
  - Available for free, and also most popular.

- Eclipse
  - Familiarity with this IDE. I've only used eclipse-based IDE.

- IntelliJ
  - Integration with language features and (for most of the time) does what it says on the tin.
- Eclipse
  - Free & extensible

- Eclipse
  - most common IDE, easy to use and has many plugins readily available

- Eclipse
  - Used and preferred by the company.

- Eclipse
  - Mature, solid, good selection of related plugins, intuitive.

- Eclipse
  - It's free and very flexible. A lot of cool plugins.

- Delphi
  - Faster results

- IntelliJ
  - simply from company

- Eclipse
  - love it

- Visual Studio
  - We actually use about 6 different languages... Power Builder, ASP, ASP.NET, C#, Java, Coldf

- Eclipse
  - Company products: Rational software architect, Websphere Integration developer.

- Coda
  - Because it brings the awesome.

- REALBasic
  - This is the IDE for the language I use.
  - Otherwise, BBEdit / vi for any PHP/web related work.

- Eclipse
  - It is free, is an industry standard for Java, it has a full set of features and has numerous plugins which can be used to assist development

- Eclipse
  - Open source. Nice CVS syncing

- IntelliJ
  - Awesome refactoring. It just works, and I don't need random plugins to get it into a working state.

- Netbeans
  - C++, Php and Java are my primary languages at the moment and Netbeans supports them all and i don't have to switch ides between projects

- Cache Studio
  - IDE supplied by vendor

- Netbeans
  - It's the chosen IDE by the senior developers and other members of the company. Project development is moving towards Java, the company is a Sun partner and Netbeans (and the rebranded Sun Studio version) will be the IDE used to develop new Java as well as C++ aspects of the product we develop.

- Visual Studio
  - Easiest interface to use

- Visual Studio
  - I was introduced to this IDE in my second year at UTS and it was such a step up from simply coding in a text document then compiling, that I've never looked back. Also it has an integration component with TortoiseSVN, a freeware SVN tool, that ensures all people in the dev. team are using the most recent code.

- Visual Studio
  - Industry standards

- none
  - Don't use an IDE

- Visual Studio
  - Supports the largest range of languages i use

- Eclipse
  - I use mostly Eclipse and Adobe Flex Builder 3 (actionscript/flex ide from adobe. it is also based on eclipse). I work mostly in actionscript and java, which fits well with the ide. eclipse is free.
  - it has subclipse, appengine, android plugins, etc.

- Programers notepad
  - open source, and does what I need (syntax colouring, compilation ext...)
- IntelliJ
  - easy and provided by company

- Collection of IBM Mainframe development tools
  - Mandatory in the company. We are an internal development department (1000 - 1500 IT developers) in the bank. Typical IBM installation

- Expresso
  - It's lightweight and easy to use. It has all the features I need without all the bloat of some of the other software.

- Visual Studio
  - Because it provides all of the functionality that I need.

- VIM and Textmate mostly
  - Both are very flexible text editors that can be configured easily and adapted to fit a variety of languages and VCSs.

- Eclipse
  - it's efficient, powerful and boosts my productivity. I don't have to spend as much time on mundane tasks.

- TextMate
  - Outstanding third-party bundles and great code completion support

- Delphi
  - Because our main application is developed in Delphi

- Visual Studio
  - We are doing .Net development. It's the best IDE for .Net.

- Eclipse
  - If I'm working in Java then I use Eclipse (I use Eclipse because when learning Java it was the first IDE I was introduced to), but if I'm working on Python, PHP, C, or something else I prefer either VI or Notepad++.

- IntelliJ
  - Great refactoring, not a lot of bugs (relatively speaking), easy to navigate and use, good code completion and other handy time-saving tools.

- Eclipse
  - it was what the team was most familiar with

- Eclipse
  - PHP, Python and subversion integrated.
  - I also use PSPad for personal development.
  - Bespin bugs.

- Visual Studio
  - Team efficiency. Availability of plug-ins (especially to support SCRUM). Optimal for .NET development.

- Bespin
  - Accessable anywhere. Will be developing into an even stronger contender in the years to come.

- Bespin
  - Because the ability to access it from any web-enabled terminal in the world has changed my entire outlook on keeping downtime to a minimum.

- Eclipse
  - Number of Plugins, Java and other Language Support, it's Open Source

- scribes
  - Simple & smart: stuff like fast code completion etc.

- Netbeans
  - works with php w/debugging of javascript and php.

- N++
  - light and fast to load/unload.
  - macros ftw.

- Eclipse

    - Free and have lots of previous experience. All the plugins are useful too.

- Bespin
    - Accessible everywhere without setup

- Eclipse
    - I did not look into Netbeans. If IntelliJ is free, I would have chosen that instead. My guys are using Eclipse as well, so that we have a unified IDE and tools (code formatting, check style, etc.).

- Eclipse
    - It is what I was taught to use, and it is used widely in the industry and there is a good amount of support available for it.

12.a) What would you change with your existing IDE?
in response to 12.) My current IDE performs all the tasks I expect of it:

- Eclipse
- Strongly Agree
    - Reduce clutter :-)

- Eclipse
- Agree
    - Better integration with git, increased speed, better communications support

- UltraEdit
- Disagree
    - in string highlighting of brackets. Better project management by using folders and drag&drop from the filesystem. Make no difference between open files and project files.

- Netbeans
- Agree
    - Still kinda bulky
    - Improved collaboration

- Netbeans
- Agree
    - NetBeans: nothing
    - Eclipse: more help figuring out how to do things

- Eclipse
- Agree
    - Performance and reliability. However, I'm not running on the latest version, so there may have been some improvements there anyway.

- Delphi
- Strongly Agree
    - Support native compilation for Mac

- Eclipse
- Strongly Agree
    - None

- Eclipse
- Neutral
    - Increased simplicity for tasks and context menus.

- Eclipse
- Strongly Agree
    - Improve speed if possible.

- Eclipse
- Agree
    - More development plugins.

- Delphi
- Agree
    - Faster, easier-to-use help system.

- IntelliJ
- Neutral
    - well documentation of error as intellije idea is full of shit in its documentation

- Eclipse
- Agree
    - Better JavaScript tools integrated in

- Visual Studio
- Strongly Agree
    - I am a DBA, so I use Toad. :-\ Not sure if this completely invalidates the entry.

- Eclipse
- Neutral
    - Make it faster!!!!!

- REALBasic
- Strongly Agree

    - Nothing at the moment

- Eclipse
- Agree
    - Better support for languages apart from Java.

- Eclipse
- Agree
    - Better PHP support

- IntelliJ
- Agree
    - Better performance. I am using a 2 quad cores + 9GB of RAM + a SSD to get it humming.

- Netbeans
- Agree
    - More flexibility with language colouring

- I switch IDEs regularly depending on the project
- Agree
    - N/A

- Cache Studio
- Neutral
    - Only has basic testing support

- Visual Studio
- Agree
    - adaptable bars

- Visual Studio
- Agree
    - Improve the help system, currently it explains what a particular function does only in terms an experienced developer would understand, also the examples either very limited or obscure, making them often difficult to assist with your issue.

- Visual Studio
- Strongly Agree
    - Better support for files not part of the project

- Eclipse
- Strongly Agree
    - hope that flex builder can be free.

- IntelliJ
- Neutral
    - documentation so that people can reference to it

- Collection of IBM Mainframe development tools
- Strongly Disagree
    - Basically most of it. The integration with the testenvironment it almost non-existent

- VIM and Textmate mostly
- Disagree
    - Better collaboration support... easier process for configuration

- Eclipse
- Agree
    - I would like to see automatically configured version control, source code sharing and concurrent editing capabilities.

- TextMate
- Agree
    - Some way of notifying if someone else has opened the same file for editing.

- Visual Studio
- Agree
    - Better integration with certain tools like code coverage.

- IntelliJ
- Agree
    - Bugfixes! IDE crashes kill productivity. I'd like better support & integration with some of the tools I use.
    - There are three strong IDEs for Java, so each has their pros and cons.
    - Also, it's not cheap.

- Visual Studio
- Strongly Agree
    - Not much. Very satisfied.

- Bespin
- Agree
    - Supplier for more in and out transport methods like FTP. More features like multiple document editing, and support for more languages.

- Bespin

- Agree
    - Ease of Installation

- Eclipse
- Agree
    - reorganization of toolbar layout, new toolbars to simply certain operations. faster
    - scribes
    - Disagree
    - Make it more of an IDE:
    - Tabs
    - Projects
    - Compile from editor
    - Built in debug
    - Currently trying Wing IDE to see if ti fits my requirements

- Netbeans
- Strongly Agree
    - more plugins

- N++
- Agree
    - I would want a better integratation to ftp and scm (i use svn).
    - ie: one file tree that allows me to browse files, show me the scm status, and ftp syncing.

- Visual Studio
- Neutral
    - better file management, support more TDD frameworks, support SVN or other popular source-control systems

- Bespin
- Disagree
    - Debugging features

- Eclipse
- Agree
    - Additional plugins for Spring, Hibernate, etc. libraries integration.

- Eclipse
- Agree
    - Project source code should be more easier to transfer from one machine to another without effecting anything else. It should work without needing to do a huge amount of configuration changes.

13a). Please describe any tools you find facilitate communication among others during development

in response to 13.) I regularly seek help from others during development

- Code review apps.
- IM, email, telephone, face-to-face communication
- Personal communication, IM, Bugzilla, Email
- IM, email, direct conversations
- msn
- Forums, Twitter, Email, IM
- email, forums
- IM, chat, email lists
- IM
- email
- Forum, IM
- Wiki, IM, Forums, Skype
- Any kind of communication tools with IM or Skype (verbal) leading the way
- human interaction, msn, skype, email
- Instant Messaging, Forums, Emails
- For direct communication - IM, email, phone calls, forums.
- For indirect communication - read others blogs, wiki snippets etc.
- Mailing list, direct Email, Google, Wikis, Forums, Blogs, FAQs
- IM and google
- Email, IM
- Phone, MS Communicator, Oracle Service Request... lol
- Sametime (IM), email
- It is all done through talking since we are in the same room. If there are bigger issues where we contact the REALBasic developers, we use email and remote debugging facilities.
- IM and (public) forums are a major part of seeking help during development
- Screen sharing
- IM
- Blogs, IM, Google
- For any non-trivial problem I find that anything but face to face communication is insufficient to properly solve a problem.
- Face-to-Face communication, Code review
- Email for sending code snippets and instructions, otherwise we talk face to face or over the phone.
- forums
- Face to face talking.
- Google to search for api's, and forums
- face-2-face communication. whiteboard.
- IM and google
- Sharepoint, email, Wiki

- mail
- IM, email, face-to-face chat, white board
- email, forums
- IM, email.
- Skype, primarily. Voice or IM.
- IMs mostly, followed by email
- Visual Studio itself, email, IM, SharePoint for more expansive project issue management.
- Forums, groups, mailing lists, wikis, and email
- Email, IM, Forums
- stackoverflow.com
- google
- im, email
- i google.
- E-Mail, Forum
- Instant messaging and email

16a.) What is the most effective tool you use for communication?
in response to 16.) My team manages communication effectively

- Email, chat.
- Face to face interaction in the same room.
- IM
- IM
- meetingroom
- IM
- email
- Maybe wiki, but it may actually be IM
- Verbal comm
- wiki
- email
- Wiki, IM - but person-to-person calls greatly help convey information - sometimes better than with IM.
- human interaction
- Face-to-face communication.
- direct - skype
- indirect - confluence
- Blogs and Wikis.
- wiki
- IM
- email
- We usually have conflicting priorities that lead us to combative disagreements. I.E. Developer has dead line, doesn't care about my database security being retro fitted to the application by the CISO.

- Email
- Sitting next to each other
- IM definitely. It is not necessary the tool that is letting us down but rather the culture of the people
- Face to face / IM / Email
- Standups, IM + Email
- IM, over the shoulder, email, skype
- Instant messaging/Voip combined with document collabrating tools.
- Face to face communication.
- cell phone and e-mail
- Either an IM client or Microsoft's SharedView, which has a built in IM and it allows users to share documents from their desktop openly with all members added to the session.
- Face to face
- Emails
- face-to-face communication
- IM
- Would not say there is one that is most effective. A lot makes it effective: email, wiki, daily standups
- telephone and mail
- IM is most frequent, but face-to-face most effective
- email, forums.
- Shout down the hall.
- Mail, feed back system
- Scrum style daily stand up meetings, and email.
- Skype, voice + IM + screen sharing.
- IMs
- IM.
- Forums/groups
- Phone, Working together (in person)
- Thinking in my brain :)
- talking
- email is all we use atm between dev members. some of us use skype to communicate with customers.
- E-Mail
- Jabber conference room, via Pidgin. Server uses OpenFire.
- Email
- issue tracker

20a.) Do you use social networking tools to facilitate development? If yes, please describe how and why you use them.

in response to 20.) Development teams benefit from the use of social and collaborative tools during project delivery

- Agree
  - Bookmarking - helps to increase development team knowledge... and bookmarks come from anyone who reads.

- Agree
  - No

- Agree
  - no

- Strongly Agree
  - We're using GitHub for our open source work and encourage a development style where every participant makes rough proposals at most and then just starts implementing changes in their fork. Integration happens on a case-by-case basis, strictly based on the solutions' merits and without a benevolent dictator model.

- Agree
  - IM's are useful to pass around url's and interleave dev work

- Agree
  - Yes, Twitter keeps me most up to date with current technology. Best news source for me.

- Agree
  - incidental use

- Agree
  - Hmmmm, not really. StackOverflow might be the closest we come.

- Strongly Agree
  - Wiki - for documentation, review and visibility purposes

- Agree
  - We currently use the status updates available in Confluence to update the team on what is ongoing on this particular project. Company wide - we use Yammer to fulfil the same goal but on a wider level.

- Agree
  - Not at this time.

- Strongly Agree
  - Not so much social networking tools as collaborative tools.

- Agree
  - no

- Disagree
  - SharePoint (our document management system) is slow, bulky and we only post things there b/c we have to. The documents are hard to work with, and disorganized. Often time there is more than one department doing the same work unbeknownst to the other.

- Agree
  - No social networking tools (other than wikis) but undobtedly a more open twitter style (what are you doing know) could be interesting for understanding what otehrs are doing AND what they have donwe in the past

- Strongly Disagree
  - No.

- Agree
  - Confluence, JIRA/greenhopper are used for development communication

- Agree
  - MSN

- Agree
  - Wiki, usually for design work.

- Agree
  - No

- Agree
  - Shared workspace to upload technical documentataion

- Neutral
  - sometimes

- Strongly Agree
  - No, but I can see the advantages / disadvantages that using them would provide. Such as different perspectives on the same issue, often result in a better executed solution. However if your audience isn't filtered to be your target audience you could start adding in problems which would otherwise be considered "out of scope".

- Neutral
  - N/A

- Disagree

    - No, dont use social networking tools like twitter or facebook. Too easy to go off track as to the topic on hand.

- Agree
    - N/A

- Neutral
    - We do not use them

- Neutral
    - not so much, but perhaps we should

- Strongly Agree
    - IM. For short, low priority discussions.

- Neutral
    - "collaborative", yes. "Social Networking". We tried "Yammer" (workplace Twitter) for a while, but it didn't go far.

- Strongly Agree
    - MS collaboration tool and sometimes adobe. MS to keep everything explained well enough to trade documents and source code. Adobe is used to present ideas back to the government leadership.

- Strongly Agree
    - Wiki's (in SharePoint), IM, Online conferencing. Speed of decision making, full team collaboration on issue exploration.

- Agree
    - Not yet

- Agree
    - twitter but mostly to goof off

- Agree
    - I want to agree with this statement, I do believe sharing and communicating more information on a daily basis would help grow the team members, but currently we dont use any social tools. I gotta get out my seat to help the chap next to me.

- Agree
    - twitter

- Neutral
    - Some of us use Twitter/Delicious to share links.

- Neutral
    - I don't use social networking tools, but I think if anything they will be not efficient as they use up a fair amount of time.

- Agree
    - wiki for documentation

22a.) Describe the impact this has on late or unsuccessful project
in response to 22.) Code conflicts occur regularly in my team

- Agree
    - The disagreement I mean here is really with implementation details found in code reviews.

- Strongly Disagree
    - Never seen it happen

- Disagree
    - We are a successful project, and always ship on time. :-)

- Neutral
    - Mostly, the merges are easy, I never experienced real delays caused by merging problems.

- Strongly Disagree
    - none

- Neutral
    - SVN usually helps us iron out most conflicts, not too much of any issue for us.

- Agree
    - Maybe I've misunderstood what you mean, but there are regular "heated" discussions between people about implementation details. I see it as healthy, frankly. But maybe you meant code conflicts in terms of integration issues? IF so, yes, they occur, but we do multiple daily builds so we find out about them quickly.

- Strongly Disagree
    - Packaging issues, possible bugs and no peace of mind

- Disagree
    - Due to the relatively few developers I am currently working with - code conflicts do not have a major impact.

- Neutral

  - Yet to have this happen so cannot answer.

- Disagree
  - It occurs and is inevitable on large scale development teams but is managed appropriately.

- Disagree
  - Affects design.

- Strongly Agree
  - Often times this means the upper management must get involved to solve the conflicting priorities.

- Neutral
  - At times it does. Especially CSS issues.

- Strongly Disagree
  - No affect, this does not happen.

- Strongly Disagree
  - Coding styles are discussed prior to program creation

- Agree
  - Due to the fact that they occur regularly, we attempt to solve this issue by setting our own deadline a few days behind the actual deadline and lock the code from changes once that date is reached. This leaves the last days for addressing any outstanding conflicts and issues, and allows for a quick review to determine if we have met our requirements.

- Agree
  - Extra time spent on re-written code

- Disagree
  - N/A. I would say what impacts projects being late/tight is bad scheduling/planning/resourcing

- Neutral
  - It's happened from time to time but to no ill effect, just some additional work to sort things out

- Disagree
  - we're not on too much of a tight schedule

- Neutral

    - If more than one person are working in the same code, there is no excuse for them not to be working together -- like, actively pair programming, or at least on the phone and well aware of what the other person is changing.
- Anything else, and your development environment is a fiasco anyway.

- Disagree
    - Code conflicts are found during the quality assurance step and depending on the nature of the fowl-- can make a delivery late.

- Disagree
    - We have minimized the conflicts be effectively exploiting continuous integration with Visual Studio.

- Disagree
    - our team is too small to have regular code conflicts - this doesnt affect the project.

- Disagree
    - none

- Strongly Agree
    - Refactoring can be a PITA when you want to have features to be pushed in the same iteration. Lesson - refactor before or after new code being added.

- Neutral
    - It makes it hard to update code that is hard to understand

24a.) What are your predictions on the likelihood of mainstream cloud-based IDEs in the near future? *followed by* 24b.) Discuss any barriers you predict may prevent this from happening

in response to 24.) I would prefer to have my IDE hosted in the cloud, rather then on my workstation

- Disagree
    - To depend on the network for work is putting the work at the risk of network failure, latency. However, they would be very popular if they support working offline.
    - Browser stability problems - I don't want to lose work when I'm using it code while at the same time have 30+ tabs open for other purposes.

- Strongly Disagree
    - It seems highly unlikely, and like a very bad idea.
    - Security concerns * People are embarrassed by their initial code * People want to test their code on their machine before they push it to the cloud * Cloud services usually run in browsers, which provide a much less rich, robust, and usable interface.

- Neutral
  - I think it is likely that you'll see cloud-based IDEs within the next few years. We're working on it, too...
  - There is a huge investment in current (desktop client) tools and plug-ins. All of this would have to be rewritten with a client/server split in mind, which is going to take a lot of time and effort.

- Disagree
  - Bespin looks promising, but has a way to go before it can be considered competition for traditional IDEs. Until I see more of that, I'm skeptical.
  - tradition - speed disadvantage of the browser (obviously less of a problem than it used to be, but still) - less integration into the OS

- Agree
  - It will happen
  - It should solve the online/offline problem.

- Strongly Agree
  - Bespin looks cool it's just not robust enough yet for our needs, plus it only hosts open source projects. I think it likely and maybe it will be a combination of a desktop and webapp. I personally am interested to see how Bespin and others develop.
  - Code security, truly Desktop rich experience of IDE on the web seems still a bit in the future.

- Disagree
  - Niche market
  - Network access is simply not as pervasive as people think it is.

- Neutral
  - In the near future, I would say the likelihood of any of them becoming mainstream is not high, purely based on the time required to get any of them to a level of maturity comparable to existing offline IDEs.
  - I think general trust of 'the cloud' will be an initial barrier, but this will reduce over time. The main barriers will be functionality. "Access anywhere" is a handy feature, but nowhere near as handy as code completion, refactoring and a bunch of other features that exist in most modern IDEs, but are yet to be seen in any cloud-based ones.

- Neutral
  - For some domains (building cloud-based systems, web development, etc) I can see that this is inevitable.
  - The barriers are not really technical to be honest. They are value. I can see the value for someone needing to develop from multiple machines, or

maybe where they are actually developing cloud-based systems, but for our line of work, the value proposition becomes less clear.

- Strongly Disagree
  - Maybe i'm just old fashioned or maybe my internet connection isn't fast enough. Just imagining the lag time gives me a headache. But i'm sure someone is going to try to do it.
  - Lag. my workstation IDE lags occasionally already. I don't want to so this kind of work which requires fast response to keep up with me on the cloud.

- Strongly Disagree
  - Unlikely to pickup very soon
  - The worldwide internet connection is not improving as much as the PC technology. Unless the connection became really fast, most developers will still prefer PC-based IDE.

- Agree
  - Possible. Loving the idea of it
  - Internet speed

- Disagree
  - unlikely. local IDE > web based.
  - connection speed, browsers, inability to do any work without internet connection

- Disagree
  - Unsure.
  - Speed of the internet, corporate IDE ownership issues, patents etc

- Disagree
  - I know about bespin and where they are going with that, remains to be seen how good the uptake will be. I would say i'm sceptical/doubtful it will make the 'mainstream' in the next 3 years (near future).
  - How embedded other IDE's are and the level of satisfaction people have with current offerings. The level new offerings need to get to before they become viable as competitors.

- Agree
  - Very likely.
  - Corporate bureaucracy in terms of security and intellectual property.

- Agree
  - Likely for simple small projects, or ones that utilize only a small stak of technoligies (e.g. a ajva appp, a c++ app). However, more complex

projects with a number of modules, custom builds and libraries would be difficult to adapt to a cloud architecture.
- Internet speed AND reliability = 1 Culture shift and perception (both individual and company level) = 2 Technology (as in better software/IDEs) = 3

- Strongly Disagree
  - It would not appeal to me, since compiling desktop apps through the cloud would use a lot of traffic and take too long, but for other languages, if it works just like a desktop application, then it would be fine. There are currently limitations to porting desktop apps to the web. (This is my iPhones now have all their apps rather than just web based applications).
  - As from above.

- Strongly Disagree
  - If net goes down.....so does development. Can't work on things while travelling (internet is expensive)
  - Need to be connected all of the time. Consume bandwidth

- Neutral
  - No change
  - potential network latency. Potentially poor performance due to the poor Javascript ;)

- Disagree
  - Not possible in my work environment.
  - All development is done in the one small office. Security issues also prevent internet/cloud based IDE solutions being used in this situation. The internet is used in development as a research tool, accessing documentation, APIs, etc.

- Neutral
  - haven't thought about it
  - connection issues

- Strongly Agree
  - Cloud-based solutions are becoming more and more prevalent with our increase in technology status and understanding. Already you can basically put your entire computer's files into a program contained in the cloud and then access them anywhere, anytime. Since many programs are starting to utilise this advantage, eg. SVN clients, I see no reason why a cloud-based IDE would not be developed and released within the next 1 - 2 years.

  - As cohesion and collaboration between different team member's and their coding increase, the ability of software to effectively maintain copywrite protection and intellectual property specifically to the individual developers decreases. This I feel will be the greatest barrier between cloud-based IDEs moving from "an interesting idea" to "mainstream IDEs, that everyone is willing to use".

- Neutral
  - 2 options: 1. Only the files are kept in the cloud, but you still use the power of the PC to do the hard work. If this is the case then the IDE doesn't have to be highly integrated with the cloud, just data access. 2. Only those who can afford it. It would be one more price tag on development that smaller shops wouldn't go for since the advantages that exist wouldn't advantage them (as there development would probably all be in house anyway, there would be no need for the access everywhere).
  - Money. 2. Current structure of who / how development is already done.

- Strongly Disagree
  - Cloud based IDEs will depend on companies paying for the extra cost of server time. Great idea until the security, IP and other issues can be solved. Not be a mainstream system for at least next 5-10 years, if it gets going.
  - Cloud IDEs are great in theory, but ownership rights, security(not only storage but location of access (internet cafes)) and other issues will impair the expansion of Cloud based IDEs. Specially with the competitive nature of system development. Cost will also be a major factor as the companies will need to expand there internal network systems specifications.

- Disagree
  - i dont like cloud-based ide... as in my country.. internet connection is very slow and it slows down my work
  - internet connection

- Neutral
  - N/A
  - N/A

- Strongly Disagree
  - They are very likely and do exist (see http://ajaxian.com/archives/heroku-web-based-rails-hosting) I am not sure how fast they will be adopted.
  - Developers like to have the power and flexibility of what is on their development machine. This means control over the file system and processor, use of multiple IDEs and apps for one single project. The All in

one IDE solution (especially for web apps) is used less and less. It's more about the right tool for the job.

- Strongly Agree
  - low - the type of cloud computing ide development now is not scalable or easily sharable. Many issues need to be worked out before cloud-based IDEs go mainstream.
  - security, compute intensive work, intermittent connectivity, cost of hosting

- Neutral
  - Bespin is certainly an impressive start, so I would say the likelihood of having cloud-based IDEs in the near future is 100%, but for them to go mainstream in the near future, 20% because I think it'll take some time for these projects to really gain traction in the mainstream
  - Comfort and familiarity. Plus, if it isn't broken, why change it?

- Strongly Agree
  - I think they will definitely appear, and some already are (Bespin, Google App Engine).
  - They have a lot of catching up to get to the point of extreme usability that say, the combination of Visual Studio and R# has. Right now they are little more than syntax highlighting text editors.

- Neutral
  - Near future? I don't think it's too near. There are advantages to cloud IDEs, in configuration time (potential ease of setup), performance, and mobility (to be able to program on a netbook or to pull up code to debug an issue while out and about -- sounds lovely). But none of these are huge wins over the current standalone IDE paradigm. It would take something like the widespread adoption of the Google Chrome OS (lightweight, web-based OS) to really push the paradigm shift.
  - Developers tend to be independent, computer-savvy, somewhat opinionated (a fair number of idiosyncratic personalities in this field) and distrustful of management. So cloud computing could be seen as a removal of independence ("what, I'm not allowed to run my own IDE?") or as management exerting control. This perception issue is the biggest barrier toward cloud IDE adoption. I believe that, for cloud IDEs to take off, developers will have to retain some semblance of control and independence

- Strongly Agree
  - Cloud computing will strenghten our ability to perform quality assurance of the code product, it will also allow enough knowledge to be shared so the maintenance cost of the customized code goes down, and (for public

sector) provides a Total Lifecycle view of the enterprise with which this product will be integrated.
    - For public sector: security in the cloud is a top priority. The fear of something catastrophic hitting the cloud and thus forcing the Government to shut it down for a spell-- is worrisome.

- Neutral
    - I don't think it needs to be an either/or approach. the desktop IDE and the (emergent) cloud IDE should be able to work in synvh with each other and exploit the advantages each offers.
    - Bandwith availability. Reticience of organizations to fully commit to cloud based solutions.

- Strongly Agree
    - Cloud-based IDEs are goi g to become extremely popular, especially if you also count locally hosted versions.
    - Concerns about network and IP security.

- Strongly Agree
    - BESPIN!
    - Nothing will stop bespin...

- Neutral
    - I like bespin but I don't have any predictions for the future. If a good capable IDE is produced, then I will use it.
    - browser speed In editor Code compilation / interpretation and debug installing modules for a language

- Disagree
    - mmmm not the near future - atlas from 280 slides guys looks like it could sorta or something like it
    - specialized application

- Strongly Agree
    - i think bespin, which is more of an ide than wave imho, will only gain traction going forward as a viable online ide. dont know how 'mainstream' it might get. i think bandwidth and web-client speed resctrictions will delay any solid market share for a few years at least.
    - in my experience while using bespin (tried to start using it fulltime): - bandwidth latency, even the extra 1 second, can be a pain when using the tool all day. - when i tried bespin (march 2009), i found it slow to render a file with more than 500 lines. - at the time i felt bespin had a steep learning curve to understand how to create projects, integrate scm etc. - this noob wants point and click tools, not all command based.

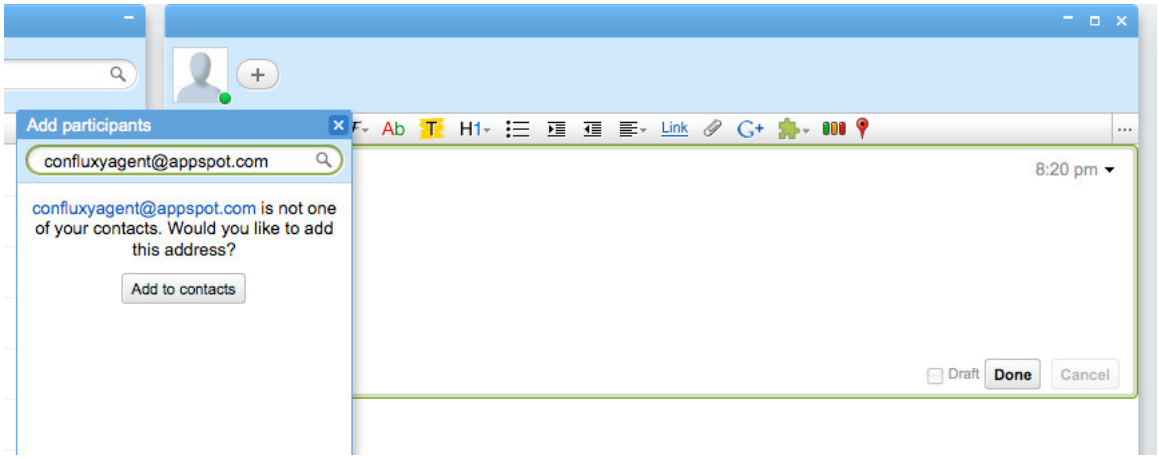
- Agree
  - This will come. However offline work and synchronization is a must.
  - Security concerns. Missing offline/sync features

- Neutral
  - I am not sure. Unlike a Word document processor (that Google Docs provides), programming is more complicated than it, e.g. code completion, rendering type hierarchy, etc. Have you seen GGE? I think it is a good start. - http://code.google.com/apis/gadgets/docs/legacy/gs.html (middle of page) - http://fusion.google.com/add?moduleurl=gge.xml
  - We need greater inet speed and robust browser.

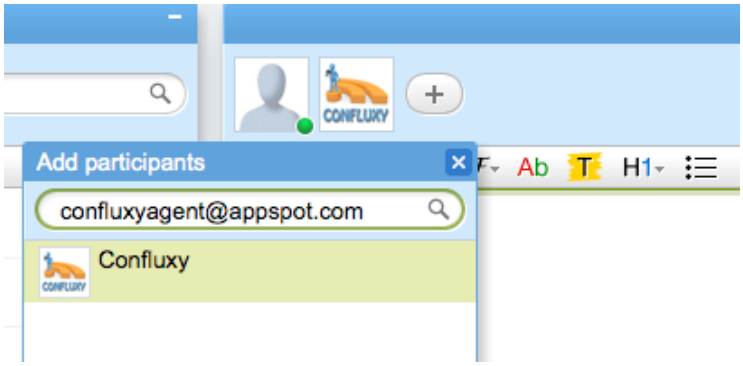
- Neutral
  - I would need to try it before I commit to it. Also it would need to become more popular in the industry and have a good amount of support available for it. Also it would need to be proven to be secure and more efficient than individual workstation IDE hosting.
  - Performance, security, network issues (increased traffic and reduced overall speed, and failures causing work to stop)

13.2. TESTING THE CONFLUXY AGENT

- Log into Google Wave at http://wave.google.com
- Invite someone into the Wave by clicking the grey plus (+) symbol on the Wave. If Confluxy is not in your contacts, the most convenience way to add it, is to simply type in its full address into the Add participants window, and click Add to contacts. Alternatively, you can manage all your contacts by clicking Manage Contacts on the bottom left area of the screen.



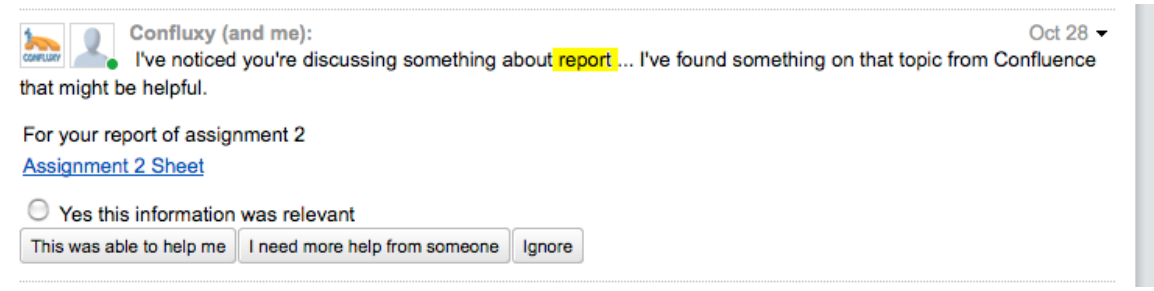
- Confluxy should appear on your participant window, simply click on it to add it to the Wave. Add more participants in the same manner if you like.



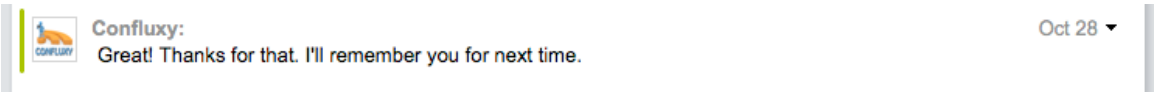
- Confluxy should tell you that it has been added to the Wave

- You can start discussing any topic you like.

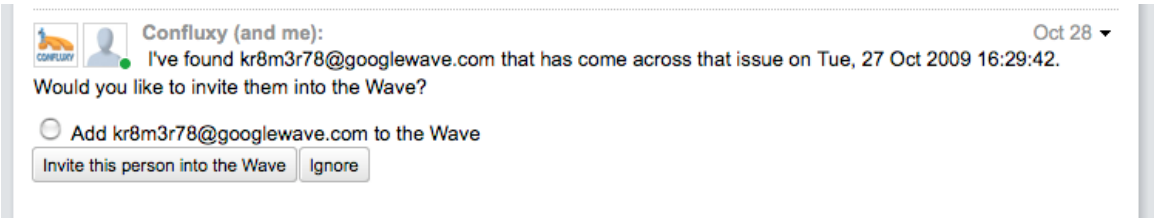
- If you mention any of the keywords Confluxy knows about, it will tell you in a new blip. If the information was helpful, click the radio button and *This was able to help me*



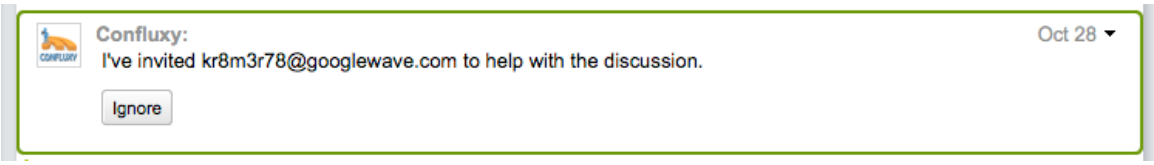
- Confluxy will now remember that you are familiar with what this topic is about for next time.



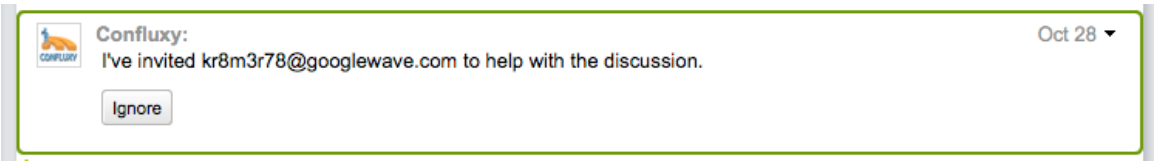
- If you selected *I need more help from someone* Confluxy will aim to look for someone that might be familiar with this topic and suggest they be added to the Wave.



- If you select the radio button and choose *Invite this person into the Wave* Confluxy will automatically invite them into the Wave to help you with the topic at hand.



- If at any point you don't want Confluxy's help, you can hit ignore and Confluxy will automatically remove the message.



- You have now tested the Confluxy agent

13.3. COMPARISON OF LITERATURE

	ALIGNMENT WITH RESEARCH				
LITERATURE	COLLABORATION	INTERNET TECHNOLOGIES	DEVELOPMENT ENVIRONMENT	INCREASED COMMUNICATION	TRANSPARENCY
Cheng et al. (2003)					
Yap et al. (2005)					
Reeves and Zhu (2004)					
Lei et al. (2004)					
Jarvensivu (2006)					

13.4. LIST OF ACCRONYMS

AJAX	Asynchronous JavaScript and XML
GAE	Google App Engine
HTML	Hypertext Markup Language
IaaS	Infrastructure as a service
IDE	Integrated Development Environment
IE	Internet Explorer
OS	Operating System
PaaS	Platform as a service
SaaS	Software as a service
URL	Uniform Resource Locator
XaaS	Anything as a service
XML	Extensible Markup Language